RsCMPX_Base

Release 5.0.60.28

Rohde & Schwarz

Apr 18, 2024

CONTENTS:


5G
CMX/P

CHAPTER

ONE

REVISION HISTORY

1.1 RsCMPX_Base

Rohde & Schwarz CMX/CMP/PVT Base System RsCMPX_Base instrument driver.

Basic Hello-World code:

```
from RsCMPX_Base import *

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

Supported instruments: CMX500, CMP200, CMP180, PVT360

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

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

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

1.1.1 Version history

Latest release notes summary: Updated for Base FW 5.0.60

Version 5.0.60

- Updated for Base FW 5.0.60, MMI SW 7.70

Version 4.0.180

- Added MMI Commands, same as in the RsCmpx-Gprf driver.

Version 4.0.175

- Fixed documentation

Version 4.0.170

- Update for FW 4.0.170

Version 4.0.140

- Update of RsCMPX_Base to FW 4.0.140 from the complete FW package 7.10.0

Version 4.0.40

- Update of RsCMPX_Base to FW 4.0.40

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

RsCMPX_Base 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

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

Preconditions

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

Option 1 - Installing with pip.exe under Windows

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

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

- Install with the command: `pip install RsCMPX_Base`

Option 2 - Installing in Pycharm

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

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

Option 3 - Offline Installation

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

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

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

- Install with the command: `pip install c:\temp\RsCMPX_Base-5.0.60.28.tar`

2.3 Finding Available Instruments

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

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

from RsCMPX_Base import *

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

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

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

from RsCMPX_Base import *

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

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

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

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

2.4 Initiating Instrument Session

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

Standard Session Initialization

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

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

- Installed RsCMPX_Base Python module Version 5.0.60 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCMPX_Base import *

# A good practice is to assure that you have a certain minimum version installed
RsCMPX_Base.assert_minimum_version('5.0.60')
resource_string_1 = 'TCPIP::192.168.2.101::INSTR'  # Standard LAN connection (also␣
↪called VXI-11)
resource_string_2 = 'TCPIP::192.168.2.101::hislip0'  # Hi-Speed LAN connection - see␣
↪1MA208
resource_string_3 = 'GPIB::20::INSTR'  # GPIB Connection
resource_string_4 = 'USB::0x0AAD::0x0119::022019943::INSTR'  # USB-TMC (Test and␣
↪Measurement Class)

# Initializing the session
driver = RsCMPX_Base(resource_string_1)

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

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

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

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

- `idn_string`
- `driver_version`
- `visa_manufacturer`
- `full_instrument_model_name`
- `instrument_serial_number`
- `instrument_firmware_version`
- `instrument_options`

The constructor also contains optional boolean arguments `id_query` and `reset`:

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

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

Selecting a Specific VISA

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

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

from RsCMPX_Base import *

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

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

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

No VISA Session

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

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

(continues on next page)

(continued from previous page)

```
from RsCMPX_Base import *

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

More `option_string` tokens are separated by comma:

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

Shared Session

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

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

from RsCMPX_Base import *

driver1 = RsCMPX_Base('TCPIP::192.168.56.101::INSTR', True, True)
driver2 = RsCMPX_Base.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 RsCMPX_Base API Structure. If for any reason you want to use the plain SCPI, use the `utilities` interface's two basic methods:

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

You may ask a question. Actually, two questions:

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

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

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

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

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

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

from RsCMPX_Base import *

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

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

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

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

from RsCMPX_Base import *

driver = RsCMPX_Base('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 RsCMPX_Base raises an exception. Speaking of exceptions, an important feature of the RsCMPX_Base is **Instrument Status Checking**. Check out the next chapter that describes the error checking in details.

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

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

from RsCMPX_Base import *

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

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

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

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

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

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

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

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

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

2.6 Error Checking

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

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

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

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

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

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

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

See the next chapter on how to react on errors.

2.7 Exception Handling

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

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

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

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

from RsCMPX_Base import *

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

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

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

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

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

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

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

Tip: General rules for exception handling:

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

2.8 Transferring Files

Instrument -> PC

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

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

PC -> Instrument

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

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

2.9 Writing Binary Data

Writing from bytes

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

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

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

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

Writing from PC files

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

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

2.10 Transferring Big Data with Progress

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

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

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

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

from RsCMPX_Base import *
import time


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

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

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


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

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

(continues on next page)

(continued from previous page)

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

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

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

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

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

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

2.11 Multithreading

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

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

One instrument session, accessed from multiple threads

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

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

import threading
from RsCMPX_Base import *
```

(continues on next page)

(continued from previous page)

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


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

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

driver.close()
```

Shared instrument session, accessed from multiple threads

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

Try the following example:

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

import threading
from RsCMPX_Base import *


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


driver1 = RsCMPX_Base('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_Base.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 RsCMPX_Base takes care of it for you. The text below describes this scenario.

Run the following example:

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

import threading
from RsCMPX_Base import *


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


driver1 = RsCMPX_Base('TCPIP::192.168.56.101::INSTR')
driver2 = RsCMPX_Base('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 RsCMPX_Base import *

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

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

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

from RsCMPX_Base import *

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

(continues on next page)

(continued from previous page)

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

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

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

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

driver.utilities.reset()

# Close the session
driver.close()

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

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

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

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

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
from RsCMPX_Base import *

driver = RsCMPX_Base('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 AdjustStatus

```
# Example value:
value = enums.AdjustStatus.ADJust
# All values (2x):
ADJust | NADJust
```

3.2 All

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

3.3 Amplification

```
# Example value:
value = enums.Amplification.HIGH
# All values (4x):
HIGH | LOW | MAXimum | MEDium
```

3.4 BaseAdjState

```
# Example value:
value = enums.BaseAdjState.ADJusted
# All values (8x):
ADJusted | AUTonomous | COUPled | INValid | OFF | ON | PENDing | RDY
```

3.5 BoxNumber

```
# First value:
value = enums.BoxNumber.BOX1
# Last value:
value = enums.BoxNumber.NAV
# All values (9x):
BOX1 | BOX2 | BOX3 | BOX4 | BOX5 | BOX6 | BOX7 | BOX8
NAV
```

3.6 ByteOrder

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

3.7 CatalogFormat

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

3.8 CmwCurrentStatus

```
# Example value:
value = enums.CmwCurrentStatus.ERRor
# All values (6x):
ERRor | MCCNconnected | MCMW | PCINconnected | SALone | STBY
```

3.9 CmwMode

```
# Example value:
value = enums.CmwMode.GENerator
# All values (3x):
GENerator | LISTener | STANdalone
```

3.10 CmwSetStatus

```
# Example value:
value = enums.CmwSetStatus.MCMW
# All values (3x):
MCMW | SALone | STBY
```

3.11 ColorSet

```
# Example value:
value = enums.ColorSet.DEF
# All values (1x):
DEF
```

3.12 CorrResult

```
# Example value:
value = enums.CorrResult.FAIL
# All values (4x):
FAIL | IPR | NCAP | PASS
```

3.13 DataFormat

```
# Example value:
value = enums.DataFormat.ASCii
# All values (8x):
ASCii | BINary | HEXadecimal | INTeger | OCTal | PACKed | REAL | UINTeger
```

3.14 DefaultUnitAngle

```
# Example value:
value = enums.DefaultUnitAngle.DEG
# All values (3x):
DEG | GRAD | RAD
```

3.15 DefaultUnitCapacity

```
# First value:
value = enums.DefaultUnitCapacity.AF
# Last value:
value = enums.DefaultUnitCapacity.UF
# All values (13x):
AF | EXF | F | FF | GF | KF | MF | MIF
NF | PEF | PF | TF | UF
```

3.16 DefaultUnitCharge

```
# First value:
value = enums.DefaultUnitCharge.AC
# Last value:
value = enums.DefaultUnitCharge.UC
# All values (13x):
AC | C | EXC | FC | GC | KC | MC | MIC
NC | PC | PEC | TC | UC
```

3.17 DefaultUnitConductance

```
# First value:
value = enums.DefaultUnitConductance.ASIE
# Last value:
value = enums.DefaultUnitConductance.USIE
# All values (13x):
ASIE | EXSie | FSIE | GSIE | KSIE | MISie | MSIE | NSIE
PESie | PSIE | SIE | TSIE | USIE
```

3.18 DefaultUnitCurrent

```
# First value:
value = enums.DefaultUnitCurrent.A
# Last value:
value = enums.DefaultUnitCurrent.UA
# All values (18x):
A | AA | DBA | DBMA | DBNA | DBPA | DBUA | EXA
FA | GA | KA | MA | MAA | NA | PA | PEA
TA | UA
```

3.19 DefaultUnitEnergy

```
# First value:
value = enums.DefaultUnitEnergy.AJ
# Last value:
value = enums.DefaultUnitEnergy.UJ
# All values (13x):
AJ | EXJ | FJ | GJ | J | KJ | MIJ | MJ
NJ | PEJ | PJ | TJ | UJ
```

3.20 DefaultUnitFrequency

```
# First value:
value = enums.DefaultUnitFrequency.AHZ
# Last value:
value = enums.DefaultUnitFrequency.UHZ
# All values (13x):
AHZ | EXHZ | FHZ | GHZ | HZ | KHZ | MHZ | MIHZ
NHZ | PEHZ | PHZ | THZ | UHZ
```

3.21 DefaultUnitLenght

```
# First value:
value = enums.DefaultUnitLenght.AM
# Last value:
value = enums.DefaultUnitLenght.UM
# All values (13x):
AM | EXM | FM | GM | KM | M | MAM | MM
NM | PEM | PM | TM | UM
```

3.22 DefaultUnitPower

```
# First value:
value = enums.DefaultUnitPower.AW
# Last value:
value = enums.DefaultUnitPower.W
# All values (19x):
AW | DBC | DBMW | DBNW | DBPW | DBUW | DBW | EXW
FW | GW | KW | MIW | MW | NW | PEW | PW
TW | UW | W
```

3.23 DefaultUnitResistor

```
# First value:
value = enums.DefaultUnitResistor.AOHM
# Last value:
value = enums.DefaultUnitResistor.UOHM
# All values (13x):
AOHM | EXOHm | FOHM | GOHM | KOHM | MIOHm | MOHM | NOHM
OHM | PEOHm | POHM | TOHM | UOHM
```

3.24 DefaultUnitTemperature

```
# Example value:
value = enums.DefaultUnitTemperature.C
# All values (6x):
C | CEL | F | FAR | K | KEL
```

3.25 DefaultUnitTime

```
# First value:
value = enums.DefaultUnitTime.AS
# Last value:
value = enums.DefaultUnitTime.US
# All values (18x):
AS | EXS | FS | GS | H | HOUR | KS | M
MAS | MIN | MS | NS | PES | PS | S | SEC
TS | US
```

3.26 DefaultUnitVoltage

```
# First value:
value = enums.DefaultUnitVoltage.AV
# Last value:
value = enums.DefaultUnitVoltage.V
# All values (18x):
AV | DBMV | DBNV | DBPV | DBUV | DBV | EXV | FV
GV | KV | MAV | MV | NV | PEV | PV | TV
UV | V
```

3.27 DeviceMode

```
# Example value:
value = enums.DeviceMode.M2X2
# All values (3x):
M2X2 | M4X4 | NONE
```

3.28 DeviceType

```
# Example value:
value = enums.DeviceType.NONE
# All values (2x):
NONE | Z24
```

3.29 DiagLoggigMode

```
# Example value:
value = enums.DiagLoggigMode.DETailed
# All values (3x):
DETailed | OFF | SIMPle
```

3.30 DiagLoggingDevice

```
# Example value:
value = enums.DiagLoggingDevice.ALL
# All values (3x):
ALL | DEBug | MEMory
```

3.31 DirectionHv

```
# Example value:
value = enums.DirectionHv.HORizontal
# All values (2x):
HORizontal | VERTical
```

3.32 DirectionIo

```
# Example value:
value = enums.DirectionIo.IN
# All values (2x):
IN | OUT
```

3.33 DisplayLanguage

```
# First value:
value = enums.DisplayLanguage.AR
# Last value:
value = enums.DisplayLanguage.ZH
# All values (14x):
AR | CS | DA | DE | EN | ES | FR | IT
JA | KO | RU | SV | TR | ZH
```

3.34 DisplayMode

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

3.35 DisplayStrategy

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

3.36 Execution

```
# Example value:
value = enums.Execution.CONCurrent
# All values (2x):
CONCurrent | SEQuential
```

3.37 ExpertSetup

```
# First value:
value = enums.ExpertSetup.BBG1
# Last value:
value = enums.ExpertSetup.SUW7
# All values (101x):
BBG1 | BBG2 | BBG3 | BBG4 | BBG5 | BBG6 | BBG7 | BBM1
BBM2 | BBM3 | BBM4 | BBM5 | BBM6 | BBM7 | INValid | PANY
PI1 | PI2 | PO1 | PO2 | R11 | R118 | R11Ci | R11Co
R11O | R12 | R12Ci | R12Co | R13 | R13Ci | R13Co | R13O
R14 | R14Ci | R14Co | R15 | R16 | R17 | R18 | R1CI
R1CO | R1O | R21Ci | R21Co | R21O | R22Ci | R22Co | R23Ci
R23Co | R23O | R24Ci | R24Co | R2CI | R2CO | R31Ci | R31Co
R31O | R32Ci | R32Co | R33Ci | R33Co | R33O | R34Ci | R34Co
R3CI | R3CO | R3O | R41Ci | R41Co | R41O | R42Ci | R42Co
R43Ci | R43Co | R43O | R44Ci | R44Co | R4CI | R4CO | RRX1
RRX2 | RRX3 | RRX4 | RTX1 | RTX2 | RTX3 | RTX4 | SUU1
SUU2 | SUU3 | SUU4 | SUU5 | SUU6 | SUU7 | SUW1 | SUW2
SUW3 | SUW4 | SUW5 | SUW6 | SUW7
```

3.38 ExpressionMode

```
# Example value:
value = enums.ExpressionMode.REGex
# All values (2x):
REGex | STRing
```

3.39 FanMode

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

3.40 FilterCriteria

```
# Example value:
value = enums.FilterCriteria.LOSupport
# All values (1x):
LOSupport
```

3.41 FontType

```
# Example value:
value = enums.FontType.DEF
# All values (2x):
DEF | LRG
```

3.42 HardcopyArea

```
# Example value:
value = enums.HardcopyArea.AWINdow
# All values (3x):
AWINdow | FSCReen | MWINdow
```

3.43 JoinAction

```
# Example value:
value = enums.JoinAction.CTASk
# All values (3x):
CTASk | DONE | STASk
```

3.44 LowHigh

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

3.45 MutexAction

```
# Example value:
value = enums.MutexAction.DONothing
# All values (2x):
DONothing | RELock
```

3.46 MutexState

```
# Example value:
value = enums.MutexState.LOCKed
# All values (3x):
LOCKed | NEWLocked | UNLocked
```

3.47 NameStyle

```
# Example value:
value = enums.NameStyle.FQName
# All values (3x):
FQName | LNAMe | NAME
```

3.48 OperationMode

```
# Example value:
value = enums.OperationMode.LOCal
# All values (2x):
LOCal | REMote
```

3.49 OscillatorType

```
# Example value:
value = enums.OscillatorType.OCXO
# All values (2x):
OCXO | TCXO
```

3.50 ProductType

```
# Example value:
value = enums.ProductType.ALL
# All values (6x):
ALL | FWA | HWOPtion | LHWoption | SWOPtion | SWPackage
```

3.51 RemoteTraceEnable

```
# Example value:
value = enums.RemoteTraceEnable.ANALysis
# All values (4x):
ANALysis | LIVE | OFF | ON
```

3.52 RemoteTraceFileFormat

```
# Example value:
value = enums.RemoteTraceFileFormat.ASCii
# All values (2x):
ASCii | XML
```

3.53 RemoteTraceStartMode

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

3.54 RemoteTraceStopMode

```
# Example value:
value = enums.RemoteTraceStopMode.AUTO
# All values (4x):
AUTO | BUFFerfull | ERRor | EXPLicit
```

3.55 Repeat

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

3.56 ResourceState

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

3.57 RfConverterInPath

```
# Example value:
value = enums.RfConverterInPath.RF1
# All values (4x):
RF1 | RF2 | RF3 | RF4
```

3.58 RollkeyMode

```
# Example value:
value = enums.RollkeyMode.CURSors
# All values (3x):
CURSors | VERTical | ZIGZag
```

3.59 RxTxDirection

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

3.60 Salignment

```
# Example value:
value = enums.Salignment.FAILed
# All values (6x):
FAILed | INValid | NAV | PASSed | PROGress | SKIPped
```

3.61 SalignmentMode

```
# Example value:
value = enums.SalignmentMode.FLEVel
# All values (5x):
FLEVel | IQ | LEVel | NAV | VIQ
```

3.62 ScreenshotFormat

```
# Example value:
value = enums.ScreenshotFormat.BMP
# All values (3x):
BMP | JPG | PNG
```

3.63 Segment

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

3.64 SelftestSpecMode

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

3.65 SelftestStopCondition

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

3.66 SelftestSumState

```
# Example value:
value = enums.SelftestSumState.FAILed
# All values (5x):
FAILed | INPRogress | INValid | PASSed | SKIPped
```

3.67 SignalSlope

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

3.68 SocketProtocol

```
# Example value:
value = enums.SocketProtocol.AGILent
# All values (3x):
AGILent | IEEE1174 | RAW
```

3.69 SourceInt

```
# Example value:
value = enums.SourceInt.EXTernal
# All values (2x):
EXTernal | INTernal
```

3.70 SourceIntExt

```
# Example value:
value = enums.SourceIntExt.EINTernal
# All values (3x):
EINTernal | EXTernal | INTernal
```

3.71 StatRegFormat

```
# Example value:
value = enums.StatRegFormat.ASCii
# All values (4x):
ASCii | BINary | HEXadecimal | OCTal
```

3.72 SyncPolling

```
# Example value:
value = enums.SyncPolling.NPOLling
# All values (2x):
NPOLling | POLLing
```

3.73 SyncResult

```
# Example value:
value = enums.SyncResult.DSTask
# All values (5x):
DSTask | NRDY | NSTask | RDY | TOUT
```

3.74 TargetStateA

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

3.75 TargetSyncState

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

3.76 TimeSource

```
# Example value:
value = enums.TimeSource.MANual
# All values (2x):
MANual | NTP
```

3.77 TriggerSource

```
# Example value:
value = enums.TriggerSource.EXTernal
# All values (4x):
EXTernal | FREerun | IF | IFPower
```

3.78 Type

```
# Example value:
value = enums.Type.CALibration
# All values (4x):
CALibration | FSCorrection | OGCal | UCORrection
```

3.79 UserRole

```
# Example value:
value = enums.UserRole.ADMin
# All values (5x):
ADMin | DEVeloper | SERVice | UEXTended | USER
```

3.80 ValidityScope

```
# Example value:
value = enums.ValidityScope.ALL
# All values (4x):
ALL | CLICense | FUNCtional | VALid
```

3.81 ValidityScopeA

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

3.82 ValidityScopeB

```
# Example value:
value = enums.ValidityScopeB.INSTrument
# All values (2x):
INSTrument | SYSTem
```

CHAPTER

FOUR

REPCAPS

4.1 Instance (Global)

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

4.2 BitNr

```
# First value:
value = repcap.BitNr.Nr8
# Range:
Nr8 .. Nr12
# All values (5x):
Nr8 | Nr9 | Nr10 | Nr11 | Nr12
```

4.3 Box

```
# First value:
value = repcap.Box.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 CmwVariant

```
# First value:
value = repcap.CmwVariant.Cmw1
# Values (2x):
Cmw1 | Cmw100
```

4.5 Eout

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

4.6 FileNr

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

4.7 Frequency

```
# First value:
value = repcap.Frequency.Freq1
# Values (4x):
Freq1 | Freq2 | Freq3 | Freq4
```

4.8 GpibInstance

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

4.9 HislipInstance

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

4.10 IpAddress

```
# First value:
value = repcap.IpAddress.Addr1
# Values (3x):
Addr1 | Addr2 | Addr3
```

4.11 NwAdapter

```
# First value:
value = repcap.NwAdapter.Adapter1
# Range:
Adapter1 .. Adapter5
# All values (5x):
Adapter1 | Adapter2 | Adapter3 | Adapter4 | Adapter5
```

4.12 RsibInstance

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

4.13 RxFilter

```
# First value:
value = repcap.RxFilter.Nr1
# Range:
Nr1 .. Nr10
# All values (10x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10
```

4.14 Slot

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

4.15 SocketInstance

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

4.16 Stream

```
# First value:
value = repcap.Stream.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
```

4.17 Trigger

```
# First value:
value = repcap.Trigger.Trg1
# Values (4x):
Trg1 | Trg2 | Trg3 | Trg4
```

4.18 TxFilter

```
# First value:
value = repcap.TxFilter.Nr1
# Range:
Nr1 .. Nr10
# All values (10x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10
```

4.19 VxiInstance

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

4.20 Window

```
# First value:
value = repcap.Window.Win1
# Range:
Win1 .. Win32
# All values (32x):
Win1 | Win2 | Win3 | Win4 | Win5 | Win6 | Win7 | Win8
Win9 | Win10 | Win11 | Win12 | Win13 | Win14 | Win15 | Win16
Win17 | Win18 | Win19 | Win20 | Win21 | Win22 | Win23 | Win24
Win25 | Win26 | Win27 | Win28 | Win29 | Win30 | Win31 | Win32
```

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

RSCMPX_BASE API STRUCTURE

Global RepCaps

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

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

647 total commands, 37 Subgroups, 3 group commands

Initializes new RsCMPX_Base session.

Parameter options tokens examples:

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

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

Parameters

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

static assert_minimum_version(*min_version: str*) → None

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

classmethod clear_global_logging_relative_timestamp() → None

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

close() → None

Closes the active RsCMPX_Base session.

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

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

Parameters

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

get_device_number() → int

```
# SCPI: *DEV
value: int = driver.get_device_number()
```

Queries the device number. It equals the Assigned Instrument number minus 1.

return
instrument_no: No help available

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_global_opc() → bool

```
# SCPI: *GOPC
value: bool = driver.get_global_opc()
```

No command help available

return
gopc: No help available

get_macro_enable() → bool

```
# SCPI: *EMC
value: bool = driver.get_macro_enable()
```

Enables or disables the execution of all macros that are defined for the active remote connection. Note: In contrast to SCPI specifications, macro execution is disabled by default.

return
enable: No help available

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

set_device_number(*instrument_no: int*) → None

```
# SCPI: *DEV
driver.set_device_number(instrument_no = 1)
```

Queries the device number. It equals the Assigned Instrument number minus 1.

param instrument_no
No help available

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.

set_macro_enable(*enable: bool*) → None

```
# SCPI: *EMC
driver.set_macro_enable(enable = False)
```

Enables or disables the execution of all macros that are defined for the active remote connection. Note: In contrast to SCPI specifications, macro execution is disabled by default.

param enable
Boolean value to enable or disable macro execution. In the disabled state (OFF / 0) , macros in a command sequence are not expanded. The CMX500 issues an error message: 113, Undefined header;MacroLabel.

Subgroups

6.1 Add

class AddCls

Add 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.add.clone()
```

Subgroups

6.1.1 System

class SystemCls

System 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.system.clone()
```

Subgroups

6.1.1.1 Attenuation

class AttenuationCls

Attenuation 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.system.attenuation.clone()
```

Subgroups

6.1.1.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.system.attenuation.correctionTable.clone()
```

Subgroups

6.1.1.1.1.1 Globale

SCPI Command :

```
ADD:SYSTem:ATTenuation:CTABle:GLOBal
```

class GlobaleCls

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

set(*name: str*, *frequency: List[float] = None*, *attenuation: List[float] = None*) → None

```
# SCPI: ADD:SYSTem:ATTenuation:CTABle:GLOBal
driver.add.system.attenuation.correctionTable.globale.set(name = 'abc',␣
↪frequency = [1.1, 2.2, 3.3], attenuation = [1.1, 2.2, 3.3])
```

Adds entries to an existing global correction table. Specify at least one parameter pair <Frequency>, <Attenuation>.

param name
Name of the existing correction table.

param frequency
No help available

param attenuation
No help available

6.1.1.1.2 Tenvironment

SCPI Command :

```
ADD:SYSTem:ATTenuation:CTABle[:TENVironment]
```

class TenvironmentCls

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

set(*name: str*, *frequency: List[float] = None*, *attenuation: List[float] = None*) → None

```
# SCPI: ADD:SYSTem:ATTenuation:CTABle[:TENVironment]
driver.add.system.attenuation.correctionTable.tenvironment.set(name = 'abc',␣
↪frequency = [1.1, 2.2, 3.3], attenuation = [1.1, 2.2, 3.3])
```

Adds entries to an existing channel-specific correction table. Specify at least one parameter pair <Frequency>, <Attenuation>.

param name
Name of the existing correction table.

param frequency
No help available

param attenuation
No help available

6.1.2 Tenvironment

class TenvironmentCls

Tenvironment 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.tenvironment.clone()
```

Subgroups

6.1.2.1 Spath

class SpathCls

Spath 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.tenvironment.spath.clone()
```

Subgroups

6.1.2.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.tenvironment.spath.correctionTable.clone()
```

Subgroups

6.1.2.1.1.1 Rx

SCPI Command :

```
ADD:TENVironment:SPATh:CTABle:RX
```

class RxCls

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

set(*name_signal_path: str*, *correction_table: List[str]*) → None

```
# SCPI: ADD:TENVironment:SPATh:CTABle:RX
driver.add.tenvironment.spath.correctionTable.rx.set(name_signal_path = 'abc',␣
↪correction_table = ['abc1', 'abc2', 'abc3'])
```

Assigns one or more correction tables to the TX direction or RX direction of a connection. If there is an existing assignment, the old assignment is kept and the new assignment is added.

param name_signal_path
Name of the connection

param correction_table
At least one name of a correction table. To assign several tables, use a comma-separated list of strings.

6.1.2.1.1.2 Tx

SCPI Command :

```
ADD:TENVironment:SPATh:CTABle:TX
```

class TxCls

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

set(*name_signal_path: str*, *correction_table: List[str]*) → None

```
# SCPI: ADD:TENVironment:SPATh:CTABle:TX
driver.add.tenvironment.spath.correctionTable.tx.set(name_signal_path = 'abc',␣
↪correction_table = ['abc1', 'abc2', 'abc3'])
```

Assigns one or more correction tables to the TX direction or RX direction of a connection. If there is an existing assignment, the old assignment is kept and the new assignment is added.

param name_signal_path
Name of the connection

param correction_table
At least one name of a correction table. To assign several tables, use a comma-separated list of strings.

6.2 Base

class BaseCls

Base commands group definition. 51 total commands, 5 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.2.1 Buffer

SCPI Commands :

```
STARt:BASE:BUFFer
STOP:BASE:BUFFer
CONTinue:BASE:BUFFer
DELete:BASE:BUFFer
CLEar:BASE:BUFFer
FETCh:BASE:BUFFer
```

class BufferCls

Buffer commands group definition. 7 total commands, 1 Subgroups, 6 group commands

clear(*buffer: str*) → None

```
# SCPI: CLEar:BASE:BUFFer
driver.base.buffer.clear(buffer = 'abc')
```

Clears the contents of a buffer. You get an empty buffer that you can fill with new commands.

param buffer
No help available

continue_py(*buffer: str*) → None

```
# SCPI: CONTinue:BASE:BUFFer
driver.base.buffer.continue_py(buffer = 'abc')
```

Reactivates a buffer which was deactivated via method RsCMPX_Base.Base.Buffer.stop) . The CMX500 continues writing data to the buffer.

param buffer
No help available

delete(*buffer: str*) → None

```
# SCPI: DELete:BASE:BUFFer
driver.base.buffer.delete(buffer = 'abc')
```

Deletes a buffer.

param buffer
No help available

fetch(*buffer: str*, *line_number: int*) → str

```
# SCPI: FETCh:BASE:BUFFer
value: str = driver.base.buffer.fetch(buffer = 'abc', line_number = 1)
```

Reads the contents of a buffer line. Buffer contents are stored line by line. Every query generates a new buffer line. The queries are not stored together with the results. Reading buffer contents is non-destructive. The lines can be read in arbitrary order.

param buffer
No help available

param line_number
The line number selecting the line to be read.

return
line: Returned line contents.

start(*buffer: str*) → None

```
# SCPI: STARt:BASE:BUFFer
driver.base.buffer.start(buffer = 'abc')
```

Creates and activates a buffer. If the buffer exists already, it is cleared (equivalent to method RsCMPX_Base.Base.Buffer. clear) .

param buffer
The buffer is identified via this label in all buffer commands.

stop() → None

```
# SCPI: STOP:BASE:BUFFer
driver.base.buffer.stop()
```

Deactivates the active buffer. Only one buffer can be active at a time. The buffer and its contents are maintained, but recording is paused. Use method RsCMPX_Base.Base.Buffer.continue_py to reactivate a buffer.

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:BASE:BUFFer
driver.base.buffer.stop_with_opc()
```

Deactivates the active buffer. Only one buffer can be active at a time. The buffer and its contents are maintained, but recording is paused. Use method RsCMPX_Base.Base.Buffer.continue_py to reactivate a buffer.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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.base.buffer.clone()
```

Subgroups

6.2.1.1 LineCount

SCPI Command :

```
FETCh:BASE:BUFFer:LINecount
```

class LineCountCls

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

fetch(*buffer: str*) → int

```
# SCPI: FETCh:BASE:BUFFer:LINecount
value: int = driver.base.buffer.lineCount.fetch(buffer = 'abc')
```

Returns the number of lines in a buffer.

param buffer
No help available

return
size: Number of lines in the buffer.

6.2.2 Correction

class CorrectionCls

Correction 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.base.correction.clone()
```

Subgroups

6.2.2.1 IfEqualizer

SCPI Commands :

```
INITiate:BASE:CORRection:IFEQualizer
ABORt:BASE:CORRection:IFEQualizer
```

class IfEqualizerCls

IfEqualizer commands group definition. 13 total commands, 3 Subgroups, 2 group commands

abort() → None

```
# SCPI: ABORt:BASE:CORRection:IFEQualizer
driver.base.correction.ifEqualizer.abort()
```

No command help available

abort_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:BASE:CORRection:IFEQualizer
driver.base.correction.ifEqualizer.abort_with_opc()
```

No command help available

Same as abort, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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:BASE:CORRection:IFEQualizer
driver.base.correction.ifEqualizer.initiate()
```

No command help available

initiate_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:BASE:CORRection:IFEQualizer
driver.base.correction.ifEqualizer.initiate_with_opc()
```

No command help available

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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.base.correction.ifEqualizer.clone()
```

Subgroups

6.2.2.1.1 Slot<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.base.correction.ifEqualizer.slot.repcap_slot_get()
driver.base.correction.ifEqualizer.slot.repcap_slot_set(repcap.Slot.Nr1)
```

class SlotCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.slot.clone()
```

Subgroups

6.2.2.1.1.1 RxFilter

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:SLOT<Slot>:RXFilter
```

class RxFilterCls

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

fetch(*slot=Slot.Default*) → List[CorrResult]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:SLOT<Slot>:RXFilter
value: List[enums.CorrResult] = driver.base.correction.ifEqualizer.slot.
↪rxFilter.fetch(slot = repcap.Slot.Default)
```

No command help available

Suppressed linked return values: reliability

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

return
value: No help available

6.2.2.1.1.2 TxFilter

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:SLOT<Slot>:TXFilter
```

class TxFilterCls

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

fetch(*slot=Slot.Default*) → List[CorrResult]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:SLOT<Slot>:TXFilter
value: List[enums.CorrResult] = driver.base.correction.ifEqualizer.slot.
↪txFilter.fetch(slot = repcap.Slot.Default)
```

No command help available

Suppressed linked return values: reliability

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

return
value: No help available

6.2.2.1.2 State

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:STATe
```

class StateCls

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

fetch() → ResourceState

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:STATe
value: enums.ResourceState = driver.base.correction.ifEqualizer.state.fetch()
```

No command help available

return
meas_status: No help available

6.2.2.1.3 Trace

class TraceCls

Trace 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.base.correction.ifEqualizer.trace.clone()
```

Subgroups

6.2.2.1.3.1 Gdelay

class GdelayCls

Gdelay 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.base.correction.ifEqualizer.trace.gdelay.clone()
```

Subgroups

6.2.2.1.3.2 Corrected

class CorrectedCls

Corrected 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.base.correction.ifEqualizer.trace.gdelay.corrected.clone()
```

Subgroups

6.2.2.1.3.3 Slot<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.repcap_slot_get()
driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.repcap_slot_set(repcap.
↪Slot.Nr1)
```

class SlotCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.clone()
```

Subgroups

6.2.2.1.3.4 RxFilter<RxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.rxFilter.repcap_
↪rxFilter_get()
driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.rxFilter.repcap_rxFilter_
↪set(repcap.RxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:CORRected:SLOT<Slot>:RXFilter<Filter>
```

class RxFilterCls

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

fetch(*slot=Slot.Default*, *rxFilter=RxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:CORRected:SLOT<Slot>
↪:RXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.gdelay.corrected.
↪slot.rxFilter.fetch(slot = repcap.Slot.Default, rxFilter = repcap.RxFilter.
↪Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.rxFilter.clone()
```

6.2.2.1.3.5 TxFilter<TxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.txFilter.repcap_
↪txFilter_get()
driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.txFilter.repcap_txFilter_
↪set(repcap.TxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:CORRected:SLOT<Slot>:TXFilter<Filter>
```

class TxFilterCls

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

fetch(*slot=Slot.Default*, *txFilter=TxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:CORRected:SLOT<Slot>
↪:TXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.gdelay.corrected.
↪slot.txFilter.fetch(slot = repcap.Slot.Default, txFilter = repcap.TxFilter.
↪Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.gdelay.corrected.slot.txFilter.clone()
```

6.2.2.1.3.6 Uncorrected

class UncorrectedCls

Uncorrected 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.base.correction.ifEqualizer.trace.gdelay.uncorrected.clone()
```

Subgroups

6.2.2.1.3.7 Slot<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.repcap_slot_get()
driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.repcap_slot_set(repcap.
↪Slot.Nr1)
```

class SlotCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.clone()
```

Subgroups

6.2.2.1.3.8 RxFilter<RxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.rxFilter.repcap_
↪rxFilter_get()
driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.rxFilter.repcap_
↪rxFilter_set(repcap.RxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:UNCorrected:SLOT<Slot>:RXFilter<Filter>
```

class RxFilterCls

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

fetch(*slot=Slot.Default*, *rxFilter=RxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:UNCorrected:SLOT<Slot>
↪:RXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.gdelay.
↪uncorrected.slot.rxFilter.fetch(slot = repcap.Slot.Default, rxFilter = repcap.
↪RxFilter.Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.rxFilter.
↪clone()
```

6.2.2.1.3.9 TxFilter<TxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.txFilter.repcap_
↪txFilter_get()
driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.txFilter.repcap_
↪txFilter_set(repcap.TxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:UNCorrected:SLOT<Slot>:TXFilter<Filter>
```

class TxFilterCls

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

fetch(*slot=Slot.Default*, *txFilter=TxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:GDELay:UNCorrected:SLOT<Slot>
↪:TXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.gdelay.
↪uncorrected.slot.txFilter.fetch(slot = repcap.Slot.Default, txFilter = repcap.
↪TxFilter.Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.gdelay.uncorrected.slot.txFilter.
↪clone()
```

6.2.2.1.3.10 Magnitude

class MagnitudeCls

Magnitude 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.base.correction.ifEqualizer.trace.magnitude.clone()
```

Subgroups

6.2.2.1.3.11 Corrected

class CorrectedCls

Corrected 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.base.correction.ifEqualizer.trace.magnitude.corrected.clone()
```

Subgroups

6.2.2.1.3.12 Slot<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.repcap_slot_get()
driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.repcap_slot_set(repcap.
↪Slot.Nr1)
```

class SlotCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.clone()
```

Subgroups

6.2.2.1.3.13 RxFilter<RxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.rxFilter.repcap_
↪rxFilter_get()
driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.rxFilter.repcap_
↪rxFilter_set(repcap.RxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:CORRected:SLOT<Slot>:RXFilter<Filter>
```

class RxFilterCls

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

fetch(*slot=Slot.Default*, *rxFilter=RxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:CORRected:SLOT<Slot>
↪:RXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.magnitude.
↪corrected.slot.rxFilter.fetch(slot = repcap.Slot.Default, rxFilter = repcap.
↪RxFilter.Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.rxFilter.
↪clone()
```

6.2.2.1.3.14 TxFilter<TxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.txFilter.repcap_
↪txFilter_get()
driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.txFilter.repcap_
↪txFilter_set(repcap.TxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:CORRected:SLOT<Slot>:TXFilter<Filter>
```

class TxFilterCls

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

fetch(*slot=Slot.Default*, *txFilter=TxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:CORRected:SLOT<Slot>
↪:TXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.magnitude.
↪corrected.slot.txFilter.fetch(slot = repcap.Slot.Default, txFilter = repcap.
↪TxFilter.Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.magnitude.corrected.slot.txFilter.
↪clone()
```

6.2.2.1.3.15 Uncorrected

class UncorrectedCls

Uncorrected 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.base.correction.ifEqualizer.trace.magnitude.uncorrected.clone()
```

Subgroups

6.2.2.1.3.16 Slot<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.repcap_slot_
↪get()
driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.repcap_slot_
↪set(repcap.Slot.Nr1)
```

class SlotCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.clone()
```

Subgroups

6.2.2.1.3.17 RxFilter<RxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.rxFilter.repcap_
↪rxFilter_get()
driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.rxFilter.repcap_
↪rxFilter_set(repcap.RxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:UNCorrected:SLOT<Slot>:RXFilter<Filter>
```

class RxFilterCls

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

fetch(*slot=Slot.Default*, *rxFilter=RxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:UNCorrected:SLOT<Slot>
↪:RXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.magnitude.
↪uncorrected.slot.rxFilter.fetch(slot = repcap.Slot.Default, rxFilter = repcap.
↪RxFilter.Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.rxFilter.
↪clone()
```

6.2.2.1.3.18 TxFilter<TxFilter>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.txFilter.repcap_
↪txFilter_get()
driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.txFilter.repcap_
↪txFilter_set(repcap.TxFilter.Nr1)
```

SCPI Command :

```
FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:UNCorrected:SLOT<Slot>:TXFilter<Filter>
```

class TxFilterCls

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

fetch(*slot=Slot.Default*, *txFilter=TxFilter.Default*) → List[float]

```
# SCPI: FETCh:BASE:CORRection:IFEQualizer:TRACe:MAGNitude:UNCorrected:SLOT<Slot>
↪:TXFilter<Filter>
value: List[float] = driver.base.correction.ifEqualizer.trace.magnitude.
↪uncorrected.slot.txFilter.fetch(slot = repcap.Slot.Default, txFilter = repcap.
↪TxFilter.Default)
```

No command help available

Suppressed linked return values: reliability

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

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

return
value: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.base.correction.ifEqualizer.trace.magnitude.uncorrected.slot.txFilter.
↪clone()
```

6.2.3 Ipc

SCPI Commands :

```
INITiate:BASE:IPC
ABORt:BASE:IPC
FETCh:BASE:IPC
```

class IpcCls

Ipc commands group definition. 4 total commands, 1 Subgroups, 3 group commands

abort() → None

```
# SCPI: ABORt:BASE:IPC
driver.base.ipc.abort()
```

No command help available

abort_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:BASE:IPC
driver.base.ipc.abort_with_opc()
```

No command help available

Same as abort, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

fetch() → ResourceState

```
# SCPI: FETCh:BASE:IPC
value: enums.ResourceState = driver.base.ipc.fetch()
```

No command help available

return
meas_status: No help available

initiate() → None

```
# SCPI: INITiate:BASE:IPC
driver.base.ipc.initiate()
```

No command help available

initiate_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:BASE:IPC
driver.base.ipc.initiate_with_opc()
```

No command help available

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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.base.ipc.clone()
```

Subgroups

6.2.3.1 Result

SCPI Command :

```
FETCh:BASE:IPC:RESult
```

class ResultCls

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

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Result_Number: int: No parameter help available
- Date: str: No parameter help available
- Result_Text: str: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:BASE:IPC:RESult
value: FetchStruct = driver.base.ipc.result.fetch()
```

No command help available

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

6.2.4 MultiCmw

SCPI Command :

```
INITiate:BASE:MCMW
```

class MultiCmwCls

MultiCmw commands group definition. 4 total commands, 3 Subgroups, 1 group commands

initiate(*cmw_1: CmwSetStatus*, *cmw_2: CmwSetStatus*, *cmw_3: CmwSetStatus*, *cmw_4: CmwSetStatus*) → None

```
# SCPI: INITiate:BASE:MCMW
driver.base.multiCmw.initiate(cmw_1 = enums.CmwSetStatus.MCMW, cmw_2 = enums.
↪CmwSetStatus.MCMW, cmw_3 = enums.CmwSetStatus.MCMW, cmw_4 = enums.
↪CmwSetStatus.MCMW)
```

No command help available

param cmw_1
No help available

param cmw_2
No help available

param cmw_3
No help available

param cmw_4
No help available

Cloning the Group

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

Subgroups

6.2.4.1 Identify

SCPI Command :

```
STARt:BASE:MCMW:IDENtify
```

class IdentifyCls

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

start(*box_nr: BoxNumber*, *blinking_time: int = None*) → None

```
# SCPI: STARt:BASE:MCMW:IDENtify
driver.base.multiCmw.identify.start(box_nr = enums.BoxNumber.BOX1, blinking_
↪time = 1)
```

No command help available

param box_nr
No help available

param blinking_time
No help available

6.2.4.2 Snumber

SCPI Command :

```
FETCh:BASE:MCMW:SNUMber
```

class SnumberCls

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

fetch(*box_nr: BoxNumber*) → str

```
# SCPI: FETCh:BASE:MCMW:SNUMber
value: str = driver.base.multiCmw.snumber.fetch(box_nr = enums.BoxNumber.BOX1)
```

No command help available

param box_nr
No help available

return
serial_number: No help available

6.2.4.3 State

SCPI Command :

```
FETCh:BASE:MCMW:STATe
```

class StateCls

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

class FetchStruct

Response structure. Fields:

- Cmw_1: enums.CmwCurrentStatus: No parameter help available
- Cmw_2: enums.CmwCurrentStatus: No parameter help available
- Cmw_3: enums.CmwCurrentStatus: No parameter help available
- Cmw_4: enums.CmwCurrentStatus: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:BASE:MCMW:STATe
value: FetchStruct = driver.base.multiCmw.state.fetch()
```

No command help available

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

6.2.5 Salignment

SCPI Commands :

```
INITiate:BASE:SALignment
ABORt:BASE:SALignment
STOP:BASE:SALignment
FETCh:BASE:SALignment
```

class SalignmentCls

Salignment commands group definition. 23 total commands, 7 Subgroups, 4 group commands

abort() → None

```
# SCPI: ABORt:BASE:SALignment
driver.base.salignment.abort()
```

Aborts the measurement procedure.

abort_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:BASE:SALignment
driver.base.salignment.abort_with_opc()
```

Aborts the measurement procedure.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

fetch() → ResourceState

```
# SCPI: FETCh:BASE:SALignment
value: enums.ResourceState = driver.base.salignment.fetch()
```

Queries the state of the measurement procedure.

return
meas_status: OFF: measurement off RUN: measurement running RDY: measurement finished

initiate() → None

```
# SCPI: INITiate:BASE:SALignment
driver.base.salignment.initiate()
```

Starts the measurement procedure.

initiate_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:BASE:SALignment
driver.base.salignment.initiate_with_opc()
```

Starts the measurement procedure.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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:BASE:SALignment
driver.base.salignment.stop()
```

Pauses the measurement procedure.

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:BASE:SALignment
driver.base.salignment.stop_with_opc()
```

Pauses the measurement procedure.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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.base.salignment.clone()
```

Subgroups

6.2.5.1 Llimit

class LlimitCls

Llimit 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.base.salignment.llimit.clone()
```

Subgroups

6.2.5.1.1 RxDc

SCPI Command :

```
FETCh:BASE:SALignment:LLIMit:RXDC
```

class RxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:LLIMit:RXDC
value: List[float] = driver.base.salignment.llimit.rxDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.1.2 RxImage

SCPI Command :

```
FETCh:BASE:SALignment:LLIMit:RXIMage
```

class RxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:LLIMit:RXIMage
value: List[float] = driver.base.salignment.llimit.rxImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.1.3 TxDc

SCPI Command :

```
FETCh:BASE:SALignment:LLIMit:TXDC
```

class TxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:LLIMit:TXDC
value: List[float] = driver.base.salignment.llimit.txDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.1.4 TxImage

SCPI Command :

```
FETCh:BASE:SALignment:LLIMit:TXIMage
```

class TxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:LLIMit:TXIMage
value: List[float] = driver.base.salignment.llimit.txImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.2 Lvalid

SCPI Command :

```
FETCh:BASE:SALignment:LVALid
```

class LvalidCls

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

class FetchStruct

Response structure. Fields:

- Date: List[str]: No parameter help available
- Time: List[str]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:BASE:SALignment:LVALid
value: FetchStruct = driver.base.salignment.lvalid.fetch()
```

Queries the date and time of the last successful execution of the self-alignment procedure. The information is returned for the measurement modes IQ and Level: <Date>IQ, <Time>IQ, <Date>level, <Time>level

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

6.2.5.3 Reliability

SCPI Command :

```
FETCh:BASE:SALignment:RELiabiliy
```

class ReliabilityCls

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

fetch() → int

```
# SCPI: FETCh:BASE:SALignment:RELiabiliy
value: int = driver.base.salignment.reliability.fetch()
```

No command help available

return
reliability: No help available

6.2.5.4 State

SCPI Command :

```
FETCh:BASE:SALignment:STATe
```

class StateCls

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

fetch() → List[Salignment]

```
# SCPI: FETCh:BASE:SALignment:STATe
value: List[enums.Salignment] = driver.base.salignment.state.fetch()
```

Queries status information for the last self-alignment execution. The information is returned for the measurement modes IQ or Verify IQ (whichever was executed last) and Level: <State>IQ or verify IQ, <State>level

return
state: No help available

6.2.5.5 Trace

class TraceCls

Trace 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.base.salignment.trace.clone()
```

Subgroups

6.2.5.5.1 RxDc

SCPI Command :

```
FETCh:BASE:SALignment:TRACe:RXDC
```

class RxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:TRACe:RXDC
value: List[float] = driver.base.salignment.trace.rxDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.5.2 RxImage

SCPI Command :

```
FETCh:BASE:SALignment:TRACe:RXIMage
```

class RxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:TRACe:RXIMage
value: List[float] = driver.base.salignment.trace.rxImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.5.3 TxDc

SCPI Command :

```
FETCh:BASE:SALignment:TRACe:TXDC
```

class TxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:TRACe:TXDC
value: List[float] = driver.base.salignment.trace.txDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.5.4 TxImage

SCPI Command :

```
FETCh:BASE:SALignment:TRACe:TXIMage
```

class TxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:TRACe:TXIMage
value: List[float] = driver.base.salignment.trace.txImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.6 Ulimit

class UlimitCls

Ulimit 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.base.salignment.ulimit.clone()
```

Subgroups

6.2.5.6.1 RxDc

SCPI Command :

```
FETCh:BASE:SALignment:ULIMit:RXDC
```

class RxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:ULIMit:RXDC
value: List[float] = driver.base.salignment.ulimit.rxDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.6.2 RxImage

SCPI Command :

```
FETCh:BASE:SALignment:ULIMit:RXIMage
```

class RxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:ULIMit:RXIMage
value: List[float] = driver.base.salignment.ulimit.rxImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.6.3 TxDc

SCPI Command :

```
FETCh:BASE:SALignment:ULIMit:TXDC
```

class TxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:ULIMit:TXDC
value: List[float] = driver.base.salignment.ulimit.txDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.6.4 TxImage

SCPI Command :

```
FETCh:BASE:SALignment:ULIMit:TXIMage
```

class TxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:ULIMit:TXIMage
value: List[float] = driver.base.salignment.ulimit.txImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.7 Xvalues

class XvaluesCls

Xvalues 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.base.salignment.xvalues.clone()
```

Subgroups

6.2.5.7.1 RxDc

SCPI Command :

```
FETCh:BASE:SALignment:XVALues:RXDC
```

class RxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:XVALues:RXDC
value: List[float] = driver.base.salignment.xvalues.rxDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.7.2 RxImage

SCPI Command :

```
FETCh:BASE:SALignment:XVALues:RXIMage
```

class RxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:XVALues:RXIMage
value: List[float] = driver.base.salignment.xvalues.rxImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.7.3 TxDc

SCPI Command :

```
FETCh:BASE:SALignment:XVALues:TXDC
```

class TxDcCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:XVALues:TXDC
value: List[float] = driver.base.salignment.xvalues.txDc.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.2.5.7.4 TxImage

SCPI Command :

```
FETCh:BASE:SALignment:XVALues:TXIMage
```

class TxImageCls

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

fetch() → List[float]

```
# SCPI: FETCh:BASE:SALignment:XVALues:TXIMage
value: List[float] = driver.base.salignment.xvalues.txImage.fetch()
```

No command help available

Suppressed linked return values: reliability

return
value: No help available

6.3 Calibration

class CalibrationCls

Calibration commands group definition. 10 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.3.1 Base

SCPI Commands :

```
CALibration:BASE:ALL
CALibration:BASE:ACFile
```

class BaseCls

Base commands group definition. 10 total commands, 3 Subgroups, 2 group commands

class AcFileStruct

Structure for reading output parameters. Fields:

- Type_Py: str: No parameter help available
- Date: str: No parameter help available

class AllStruct

Structure for reading output parameters. Fields:

- Date: List[str]: No parameter help available
- Time: List[str]: No parameter help available
- Type_Py: List[enums.Type]: No parameter help available

get_ac_file() → AcFileStruct

```
# SCPI: CALibration:BASE:ACFile
value: AcFileStruct = driver.calibration.base.get_ac_file()
```

No command help available

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

get_all() → AllStruct

```
# SCPI: CALibration:BASE:ALL
value: AllStruct = driver.calibration.base.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.calibration.base.clone()
```

Subgroups

6.3.1.1 Ipc

SCPI Commands :

```
CALibration:BASE:IPC:RESult
CALibration:BASE:IPC:VALues
CALibration:BASE:IPC:LOG
```

class IpcCls

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

class ResultStruct

Structure for reading output parameters. Fields:

- Result_Number: int: No parameter help available
- Date: str: No parameter help available
- Result_Text: str: No parameter help available

class ValuesStruct

Structure for reading output parameters. Fields:

- Min_Py: List[float]: No parameter help available
- Fmin: List[int]: No parameter help available
- Max_Py: List[float]: No parameter help available
- Fmax: List[int]: No parameter help available

get_log() → List[str]

```
# SCPI: CALibration:BASE:IPC:LOG
value: List[str] = driver.calibration.base.ipc.get_log()
```

No command help available

return
file_path: No help available

get_result() → ResultStruct

```
# SCPI: CALibration:BASE:IPC:RESult
value: ResultStruct = driver.calibration.base.ipc.get_result()
```

No command help available

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

get_values() → ValuesStruct

```
# SCPI: CALibration:BASE:IPC:VALues
value: ValuesStruct = driver.calibration.base.ipc.get_values()
```

No command help available

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

6.3.1.2 Ipcr

SCPI Commands :

```
CALibration:BASE:IPCR:DATE
CALibration:BASE:IPCR:STATe
CALibration:BASE:IPCR:RESult
```

class IpcrCls

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

get_date() → str

```
# SCPI: CALibration:BASE:IPCR:DATE
value: str = driver.calibration.base.ipcr.get_date()
```

No command help available

return
date: No help available

get_result() → List[str]

```
# SCPI: CALibration:BASE:IPCR:RESult
value: List[str] = driver.calibration.base.ipcr.get_result()
```

No command help available

return
result: No help available

get_state() → List[int]

```
# SCPI: CALibration:BASE:IPCR:STATe
value: List[int] = driver.calibration.base.ipcr.get_state()
```

No command help available

return
state: No help available

6.3.1.3 Latest

SCPI Command :

```
CALibration:BASE:LATest
```

class LatestCls

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

class GetStruct

Response structure. Fields:

- Date: str: No parameter help available
- Time: str: No parameter help available
- Type_Py: enums.Type: No parameter help available

get(*type_py: Type = None*) → GetStruct

```
# SCPI: CALibration:BASE:LATest
value: GetStruct = driver.calibration.base.latest.get(type_py = enums.Type.
↪CALibration)
```

No command help available

param type_py
No help available

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

Cloning the Group

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

Subgroups

6.3.1.3.1 Specific

SCPI Command :

```
CALibration:BASE:LATest:SPECific
```

class SpecificCls

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

class GetStruct

Response structure. Fields:

- Date: str: No parameter help available
- Time: str: No parameter help available

get(*mode: Type*) → GetStruct

```
# SCPI: CALibration:BASE:LATest:SPECific
value: GetStruct = driver.calibration.base.latest.specific.get(mode = enums.
↪Type.CALibration)
```

No command help available

param mode
No help available

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

6.4 Catalog

class CatalogCls

Catalog commands group definition. 31 total commands, 18 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.4.1 Base

class BaseCls

Base 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.catalog.base.clone()
```

Subgroups

6.4.1.1 Correction

class CorrectionCls

Correction 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.base.correction.clone()
```

Subgroups

6.4.1.1.1 IfEqualizer

SCPI Command :

```
CATalog:BASE:CORRection:IFEQualizer:SNAMe
```

class IfEqualizerCls

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

get_sname() → List[str]

```
# SCPI: CATalog:BASE:CORRection:IFEQualizer:SNAMe
value: List[str] = driver.catalog.base.correction.ifEqualizer.get_sname()
```

No command help available

return
slot: No help available

Cloning the Group

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

Subgroups

6.4.1.1.1.1 Slot<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.catalog.base.correction.ifEqualizer.slot.repcap_slot_get()
driver.catalog.base.correction.ifEqualizer.slot.repcap_slot_set(repcap.Slot.Nr1)
```

class SlotCls

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

Cloning the Group

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

Subgroups

6.4.1.1.1.2 RxFilter

SCPI Command :

```
CATalog:BASE:CORRection:IFEQualizer:SLOT<Slot>:RXFilter
```

class RxFilterCls

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

get(*slot*=*Slot.Default*) → List[str]

```
# SCPI: CATalog:BASE:CORRection:IFEQualizer:SLOT<Slot>:RXFilter
value: List[str] = driver.catalog.base.correction.ifEqualizer.slot.rxFilter.
↪get(slot = repcap.Slot.Default)
```

No command help available

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

return
filter_py: No help available

6.4.1.1.1.3 TxFilter

SCPI Command :

```
CATalog:BASE:CORRection:IFEQualizer:SLOT<Slot>:TXFilter
```

class TxFilterCls

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

get(*slot*=*Slot.Default*) → List[str]

```
# SCPI: CATalog:BASE:CORRection:IFEQualizer:SLOT<Slot>:TXFilter
value: List[str] = driver.catalog.base.correction.ifEqualizer.slot.txFilter.
↪get(slot = repcap.Slot.Default)
```

No command help available

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

return
filter_py: No help available

6.4.1.2 Salignment

SCPI Command :

```
CATalog:BASE:SALignment:SLOT
```

class SalignmentCls

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

get_slot() → str

```
# SCPI: CATalog:BASE:SALignment:SLOT
value: str = driver.catalog.base.salignment.get_slot()
```

No command help available

return
slot_list: No help available

6.4.2 Bluetooth

class BluetoothCls

Bluetooth 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.bluetooth.clone()
```

Subgroups

6.4.2.1 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.catalog.bluetooth.measurement.clone()
```

Subgroups

6.4.2.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.bluetooth.measurement.spath.repcap_stream_get()
driver.catalog.bluetooth.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:BLUetooth:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:BLUetooth:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.bluetooth.measurement.spath.get(stream =␣
↪repcap.Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.3 Cdma

class CdmaCls

Cdma 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.cdma.clone()
```

Subgroups

6.4.3.1 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.catalog.cdma.measurement.clone()
```

Subgroups

6.4.3.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.cdma.measurement.spath.repcap_stream_get()
driver.catalog.cdma.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:CDMA:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:CDMA:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.cdma.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.4 Gprf

class GprfCls

Gprf 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.catalog.gprf.clone()
```

Subgroups

6.4.4.1 Generator

class GeneratorCls

Generator 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.gprf.generator.clone()
```

Subgroups

6.4.4.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.gprf.generator.spath.repcap_stream_get()
driver.catalog.gprf.generator.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:GPRF:GENerator<Instance>:SPATh<StreamNumber>
```

class SpathCls

Spath commands group definition. 3 total commands, 1 Subgroups, 1 group commands Repeated Capability: Stream, default value after init: Stream.Nr1

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:GPRF:GENerator<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.gprf.generator.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

Subgroups

6.4.4.1.1.1 Group

SCPI Commands :

```
CATalog:GPRF:GENerator<Instance>:SPATh:GROup:CONNector
CATalog:GPRF:GENerator<Instance>:SPATh:GROup
```

class GroupCls

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

get(*connector_name: str*) → List[str]

```
# SCPI: CATalog:GPRF:GENerator<Instance>:SPATh:GROup
value: List[str] = driver.catalog.gprf.generator.spath.group.get(connector_name␣
↪= 'abc')
```

No command help available

param connector_name
No help available

return
signal_path: No help available

get_connector() → str

```
# SCPI: CATalog:GPRF:GENerator<Instance>:SPATh:GROup:CONNector
value: str = driver.catalog.gprf.generator.spath.group.get_connector()
```

No command help available

return
connector_name: No help available

6.4.4.2 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.catalog.gprf.measurement.clone()
```

Subgroups

6.4.4.2.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.gprf.measurement.spath.repcap_stream_get()
driver.catalog.gprf.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:GPRF:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:GPRF:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.gprf.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.5 Gsm

class GsmCls

Gsm 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.gsm.clone()
```

Subgroups

6.4.5.1 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.catalog.gsm.measurement.clone()
```

Subgroups

6.4.5.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.gsm.measurement.spath.repcap_stream_get()
driver.catalog.gsm.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:GSM:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:GSM:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.gsm.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.6 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.catalog.lte.clone()
```

Subgroups

6.4.6.1 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.catalog.lte.measurement.clone()
```

Subgroups

6.4.6.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.lte.measurement.spath.repcap_stream_get()
driver.catalog.lte.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:LTE:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:LTE:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.lte.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.7 LteDl

class LteDlCls

LteDl 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.lteDl.clone()
```

Subgroups

6.4.7.1 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.catalog.lteDl.measurement.clone()
```

Subgroups

6.4.7.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.lteDl.measurement.spath.repcap_stream_get()
driver.catalog.lteDl.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:LTEDl:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:LTEDl:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.lteDl.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.8 Niot

class NiotCls

Niot 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.niot.clone()
```

Subgroups

6.4.8.1 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.catalog.niot.measurement.clone()
```

Subgroups

6.4.8.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.niot.measurement.spath.repcap_stream_get()
driver.catalog.niot.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:NIOT:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:NIOT:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.niot.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.9 NrDl

class NrDlCls

NrDl 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.nrDl.clone()
```

Subgroups

6.4.9.1 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.catalog.nrDl.measurement.clone()
```

Subgroups

6.4.9.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.nrDl.measurement.spath.repcap_stream_get()
driver.catalog.nrDl.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:NRDL:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:NRDL:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.nrDl.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.10 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.catalog.nrMmw.clone()
```

Subgroups

6.4.10.1 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.catalog.nrMmw.measurement.clone()
```

Subgroups

6.4.10.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.nrMmw.measurement.spath.repcap_stream_get()
driver.catalog.nrMmw.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:NRMMw:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:NRMMw:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.nrMmw.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.11 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.catalog.nrSub.clone()
```

Subgroups

6.4.11.1 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.catalog.nrSub.measurement.clone()
```

Subgroups

6.4.11.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.nrSub.measurement.spath.repcap_stream_get()
driver.catalog.nrSub.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:NRSub:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:NRSub:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.nrSub.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.12 Selftest

SCPI Commands :

```
CATalog:SELFtest
CATalog:SELFtest:UPRofile
```

class SelftestCls

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

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

```
# SCPI: CATalog:SELFtest
value: List[str] = driver.catalog.selftest.get(filter_py = 'abc')
```

No command help available

param filter_py
No help available

return
results: No help available

get_uprofile() → List[str]

```
# SCPI: CATalog:SELFtest:UPRofile
value: List[str] = driver.catalog.selftest.get_uprofile()
```

No command help available

return
user_prof_names: No help available

Cloning the Group

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

Subgroups

6.4.12.1 Selected

SCPI Command :

```
CATalog:SELFtest:SELected
```

class SelectedCls

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

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

```
# SCPI: CATalog:SELFtest:SELected
value: List[str] = driver.catalog.selftest.selected.get(filter_py = 'abc')
```

No command help available

param filter_py
No help available

return
result: No help available

6.4.13 System

class SystemCls

System 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.catalog.system.clone()
```

Subgroups

6.4.13.1 Attenuation

class AttenuationCls

Attenuation 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.system.attenuation.clone()
```

Subgroups

6.4.13.1.1 CorrectionTable

SCPI Commands :

```
CATalog:SYSTem:ATTenuation:CTABle[:TENVironment]
CATalog:SYSTem:ATTenuation:CTABle:GLOBal
```

class CorrectionTableCls

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

get_globale() → str

```
# SCPI: CATalog:SYSTem:ATTenuation:CTABle:GLOBal
value: str = driver.catalog.system.attenuation.correctionTable.get_globale()
```

Returns the names of all global correction tables.

return
name: Comma-separated list of strings, one string per correction table.

get_tenvironment() → str

```
# SCPI: CATalog:SYSTem:ATTenuation:CTABle[:TENVironment]
value: str = driver.catalog.system.attenuation.correctionTable.get_
↪tenvironment()
```

Returns the names of all channel-specific correction tables for the addressed smart channel.

return
name: Comma-separated list of strings, one string per correction table.

6.4.13.2 Reset

SCPI Command :

```
CATalog:SYSTem:RESet:PARTial
```

class ResetCls

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

get_partial() → List[str]

```
# SCPI: CATalog:SYSTem:RESet:PARTial
value: List[str] = driver.catalog.system.reset.get_partial()
```

Queries a list of the components available for a partial reset.

return
resetable_system_part: Comma-separated list of strings, one string per component.

6.4.13.3 Rf42

class Rf42Cls

Rf42 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.system.rf42.clone()
```

Subgroups

6.4.13.3.1 Box

SCPI Command :

```
CATalog:SYSTem:RF42:BOX
```

class BoxCls

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

get(*serial_number: str*) → str

```
# SCPI: CATalog:SYSTem:RF42:BOX
value: str = driver.catalog.system.rf42.box.get(serial_number = 'abc')
```

Queries a list of all connected R&S CMX-RF42.

param serial_number
Comma-separated list of serial numbers, one string per detected R&S CMX-RF42.

return
result: No help available

6.4.13.4 Rrhead

SCPI Command :

```
CATalog:SYSTem:RRHead
```

class RrheadCls

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

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

```
# SCPI: CATalog:SYSTem:RRHead
value: List[str] = driver.catalog.system.rrhead.get(filter_criteria = enums.
↪FilterCriteria.LOSupport)
```

Queries a list of all connected remote radio heads.

param filter_criteria
Return only remote radio heads supporting external LO.

return
rrh_name: Comma-separated list of RRH names, one string per RRH

6.4.14 Tenvironment

SCPI Command :

```
CATalog:TENVironment:SPATh
```

class TenvironmentCls

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

get_spath() → List[str]

```
# SCPI: CATalog:TENVironment:SPATh
value: List[str] = driver.catalog.tenvironment.get_spath()
```

Returns all connection names.

return
name_signal_path: Comma-separated list of strings, one string per connection.

Cloning the Group

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

Subgroups

6.4.14.1 Connectors

SCPI Command :

```
CATalog:TENVironment:CONNectors
```

class ConnectorsCls

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

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

```
# SCPI: CATalog:TENVironment:CONNectors
value: List[str] = driver.catalog.tenvironment.connectors.get(name_style =␣
↪enums.NameStyle.FQName)
```

No command help available

param name_style
No help available

return
name_connector: No help available

6.4.15 Uwb

class UwbCls

Uwb 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.uwb.clone()
```

Subgroups

6.4.15.1 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.catalog.uwb.measurement.clone()
```

Subgroups

6.4.15.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.uwb.measurement.spath.repcap_stream_get()
driver.catalog.uwb.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:UWB:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:UWB:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.uwb.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.16 Wcdma

class WcdmaCls

Wcdma 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.wcdma.clone()
```

Subgroups

6.4.16.1 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.catalog.wcdma.measurement.clone()
```

Subgroups

6.4.16.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.wcdma.measurement.spath.repcap_stream_get()
driver.catalog.wcdma.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:WCDMa:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:WCDMa:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.wcdma.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.17 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.catalog.wlan.clone()
```

Subgroups

6.4.17.1 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.catalog.wlan.measurement.clone()
```

Subgroups

6.4.17.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.wlan.measurement.spath.repcap_stream_get()
driver.catalog.wlan.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:WLAN:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:WLAN:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.wlan.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.4.18 Wpan

class WpanCls

Wpan 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.wpan.clone()
```

Subgroups

6.4.18.1 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.catalog.wpan.measurement.clone()
```

Subgroups

6.4.18.1.1 Spath<Stream>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.catalog.wpan.measurement.spath.repcap_stream_get()
driver.catalog.wpan.measurement.spath.repcap_stream_set(repcap.Stream.Nr1)
```

SCPI Command :

```
CATalog:WPAN:MEASurement<Instance>:SPATh<StreamNumber>
```

class SpathCls

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

get(*stream=Stream.Default*) → List[str]

```
# SCPI: CATalog:WPAN:MEASurement<Instance>:SPATh<StreamNumber>
value: List[str] = driver.catalog.wpan.measurement.spath.get(stream = repcap.
↪Stream.Default)
```

No command help available

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

return
name_signal_path: No help available

Cloning the Group

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

6.5 Cmwd

SCPI Commands :

```
INITiate:CMWD
STOP:CMWD
ABORt:CMWD
FETCh:CMWD
```

class CmwdCls

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

abort() → None

```
# SCPI: ABORt:CMWD
driver.cmwd.abort()
```

No command help available

abort_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:CMWD
driver.cmwd.abort_with_opc()
```

No command help available

Same as abort, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

fetch() → str

```
# SCPI: FETCh:CMWD
value: str = driver.cmwd.fetch()
```

No command help available

Suppressed linked return values: reliability

return
result_string: No help available

initiate() → None

```
# SCPI: INITiate:CMWD
driver.cmwd.initiate()
```

No command help available

initiate_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INITiate:CMWD
driver.cmwd.initiate_with_opc()
```

No command help available

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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:CMWD
driver.cmwd.stop()
```

No command help available

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:CMWD
driver.cmwd.stop_with_opc()
```

No command help available

Same as stop, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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.cmwd.clone()
```

Subgroups

6.5.1 State

SCPI Command :

```
FETCh:CMWD:STATe
```

class StateCls

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

fetch() → ResourceState

```
# SCPI: FETCh:CMWD:STATe
value: enums.ResourceState = driver.cmwd.state.fetch()
```

No command help available

return
meas_state: No help available

6.6 Configure

class ConfigureCls

Configure commands group definition. 94 total commands, 11 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.6.1 Base

SCPI Command :

```
CONFigure:BASE:FCONtrol
```

class BaseCls

Base commands group definition. 27 total commands, 8 Subgroups, 1 group commands

get_fcontrol() → FanMode

```
# SCPI: CONFigure:BASE:FCONtrol
value: enums.FanMode = driver.configure.base.get_fcontrol()
```

No command help available

return
mode: No help available

set_fcontrol(*mode: FanMode*) → None

```
# SCPI: CONFigure:BASE:FCONtrol
driver.configure.base.set_fcontrol(mode = enums.FanMode.HIGH)
```

No command help available

param mode
No help available

Cloning the Group

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

Subgroups

6.6.1.1 Adjustment

SCPI Commands :

```
CONFigure:BASE:ADJustment:TYPE
CONFigure:BASE:ADJustment:VALue
CONFigure:BASE:ADJustment:SAVE
```

class AdjustmentCls

Adjustment commands group definition. 4 total commands, 1 Subgroups, 3 group commands

get_type_py() → OscillatorType

```
# SCPI: CONFigure:BASE:ADJustment:TYPE
value: enums.OscillatorType = driver.configure.base.adjustment.get_type_py()
```

No command help available

return
adj_type: No help available

get_value() → float

```
# SCPI: CONFigure:BASE:ADJustment:VALue
value: float = driver.configure.base.adjustment.get_value()
```

No command help available

return
adj_value: No help available

save() → None

```
# SCPI: CONFigure:BASE:ADJustment:SAVE
driver.configure.base.adjustment.save()
```

No command help available

save_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CONFigure:BASE:ADJustment:SAVE
driver.configure.base.adjustment.save_with_opc()
```

No command help available

Same as save, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

set_value(*adj_value: float*) → None

```
# SCPI: CONFigure:BASE:ADJustment:VALue
driver.configure.base.adjustment.set_value(adj_value = 1.0)
```

No command help available

param adj_value
No help available

Cloning the Group

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

Subgroups

6.6.1.1.1 SfDefault

SCPI Command :

```
CONFigure:BASE:ADJustment:SFDefault
```

class SfDefaultCls

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

set() → None

```
# SCPI: CONFigure:BASE:ADJustment:SFDefault
driver.configure.base.adjustment.sfDefault.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: CONFigure:BASE:ADJustment:SFDefault
driver.configure.base.adjustment.sfDefault.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.6.1.2 Correction

class CorrectionCls

Correction 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.base.correction.clone()
```

Subgroups

6.6.1.2.1 IfEqualizer

class IfEqualizerCls

IfEqualizer 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.base.correction.ifEqualizer.clone()
```

Subgroups

6.6.1.2.1.1 Slot<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.base.correction.ifEqualizer.slot.repcap_slot_get()
driver.configure.base.correction.ifEqualizer.slot.repcap_slot_set(repcap.Slot.Nr1)
```

class SlotCls

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

Cloning the Group

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

Subgroups

6.6.1.2.1.2 RxFilter

class RxFilterCls

RxFilter 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.base.correction.ifEqualizer.slot.rxFilter.clone()
```

Subgroups

6.6.1.2.1.3 Select

SCPI Command :

```
CONFigure:BASE:CORRection:IFEQualizer:SLOT<Slot>:RXFilter:SELect
```

class SelectCls

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

get(*slot=Slot.Default*) → List[bool]

```
# SCPI: CONFigure:BASE:CORRection:IFEQualizer:SLOT<Slot>:RXFilter:SELect
value: List[bool] = driver.configure.base.correction.ifEqualizer.slot.rxFilter.
↪select.get(slot = repcap.Slot.Default)
```

No command help available

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

return
select: No help available

set(*select: List[bool]*, *slot=Slot.Default*) → None

```
# SCPI: CONFigure:BASE:CORRection:IFEQualizer:SLOT<Slot>:RXFilter:SELect
driver.configure.base.correction.ifEqualizer.slot.rxFilter.select.set(select =␣
↪[True, False, True], slot = repcap.Slot.Default)
```

No command help available

param select
No help available

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

6.6.1.2.1.4 TxFilter

class TxFilterCls

TxFilter 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.base.correction.ifEqualizer.slot.txFilter.clone()
```

Subgroups

6.6.1.2.1.5 Select

SCPI Command :

```
CONFigure:BASE:CORRection:IFEQualizer:SLOT<Slot>:TXFilter:SELect
```

class SelectCls

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

get(*slot=Slot.Default*) → List[bool]

```
# SCPI: CONFigure:BASE:CORRection:IFEQualizer:SLOT<Slot>:TXFilter:SELect
value: List[bool] = driver.configure.base.correction.ifEqualizer.slot.txFilter.
↪select.get(slot = repcap.Slot.Default)
```

No command help available

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

return
select: No help available

set(*select: List[bool]*, *slot=Slot.Default*) → None

```
# SCPI: CONFigure:BASE:CORRection:IFEQualizer:SLOT<Slot>:TXFilter:SELect
driver.configure.base.correction.ifEqualizer.slot.txFilter.select.set(select =␣
↪[True, False, True], slot = repcap.Slot.Default)
```

No command help available

param select
No help available

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

6.6.1.3 FreqCorrection

SCPI Commands :

```
CONFigure:BASE:FDCorrection:SAV
CONFigure:BASE:FDCorrection:RCL
```

class FreqCorrectionCls

FreqCorrection commands group definition. 12 total commands, 1 Subgroups, 2 group commands

recall(*table_path: str = None*) → None

```
# SCPI: CONFigure:BASE:FDCorrection:RCL
driver.configure.base.freqCorrection.recall(table_path = rawAbc)
```

No command help available

param table_path
No help available

save(*table_path: str = None*) → None

```
# SCPI: CONFigure:BASE:FDCorrection:SAV
driver.configure.base.freqCorrection.save(table_path = rawAbc)
```

No command help available

param table_path
No help available

Cloning the Group

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

Subgroups

6.6.1.3.1 CorrectionTable

SCPI Commands :

```
CONFigure:BASE:FDCorrection:CTABle:DELete
CONFigure:BASE:FDCorrection:CTABle:DELete:ALL
```

class CorrectionTableCls

CorrectionTable commands group definition. 10 total commands, 8 Subgroups, 2 group commands

delete(*table_name: str*) → None

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:DELete
driver.configure.base.freqCorrection.correctionTable.delete(table_name = 'abc')
```

No command help available

param table_name
No help available

delete_all(*table_path: str = None*) → None

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:DELete:ALL
driver.configure.base.freqCorrection.correctionTable.delete_all(table_path =␣
↪rawAbc)
```

No command help available

param table_path
No help available

Cloning the Group

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

Subgroups

6.6.1.3.1.1 Add

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:ADD
```

class AddCls

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

set(*table_name: str*, *frequency: List[float] = None*, *correction: List[float] = None*) → None

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:ADD
driver.configure.base.freqCorrection.correctionTable.add.set(table_name = 'abc',
↪ frequency = [1.1, 2.2, 3.3], correction = [1.1, 2.2, 3.3])
```

No command help available

param table_name
No help available

param frequency
No help available

param correction
No help available

6.6.1.3.1.2 Catalog

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:CATalog
```

class CatalogCls

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

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

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:CATalog
value: List[str] = driver.configure.base.freqCorrection.correctionTable.catalog.
↪get(table_path = rawAbc)
```

No command help available

param table_path
No help available

return
table_name: No help available

6.6.1.3.1.3 Count

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:COUNt
```

class CountCls

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

get(*table_path: str = None*) → int

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:COUNt
value: int = driver.configure.base.freqCorrection.correctionTable.count.
↪get(table_path = rawAbc)
```

No command help available

param table_path
No help available

return
table_count: No help available

6.6.1.3.1.4 Create

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:CREate
```

class CreateCls

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

set(*table_name: str*, *frequency: List[float] = None*, *correction: List[float] = None*) → None

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:CREate
driver.configure.base.freqCorrection.correctionTable.create.set(table_name =
↪'abc', frequency = [1.1, 2.2, 3.3], correction = [1.1, 2.2, 3.3])
```

No command help available

param table_name
No help available

param frequency
No help available

param correction
No help available

6.6.1.3.1.5 Details

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:DETails
```

class DetailsCls

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

class GetStruct

Response structure. Fields:

- Frequency: List[float]: No parameter help available
- Correction: List[float]: No parameter help available

get(*table_name: str*, *start_index: float = None*, *count: float = None*) → GetStruct

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:DETails
value: GetStruct = driver.configure.base.freqCorrection.correctionTable.details.
↪get(table_name = 'abc', start_index = 1.0, count = 1.0)
```

No command help available

param table_name
No help available

param start_index
No help available

param count
No help available

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

6.6.1.3.1.6 Erase

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:ERASe
```

class EraseCls

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

set(*table_name: str*, *frequency: List[float] = None*) → None

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:ERASe
driver.configure.base.freqCorrection.correctionTable.erase.set(table_name = 'abc
↪', frequency = [1.1, 2.2, 3.3])
```

No command help available

param table_name
No help available

param frequency
No help available

6.6.1.3.1.7 Exist

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:EXISt
```

class ExistCls

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

get(*table_name: str*) → int

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:EXISt
value: int = driver.configure.base.freqCorrection.correctionTable.exist.
↪get(table_name = 'abc')
```

No command help available

param table_name
No help available

return
exists: No help available

6.6.1.3.1.8 Length

SCPI Command :

```
CONFigure:BASE:FDCorrection:CTABle:LENGth
```

class LengthCls

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

get(*table_name: str*) → int

```
# SCPI: CONFigure:BASE:FDCorrection:CTABle:LENGth
value: int = driver.configure.base.freqCorrection.correctionTable.length.
↪get(table_name = 'abc')
```

No command help available

param table_name
No help available

return
table_length: No help available

6.6.1.4 Ipcr

SCPI Commands :

```
CONFigure:BASE:IPCR:ENABle
CONFigure:BASE:IPCR:IDENt
```

class IpcrCls

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

get_enable() → List[bool]

```
# SCPI: CONFigure:BASE:IPCR:ENABle
value: List[bool] = driver.configure.base.ipcr.get_enable()
```

No command help available

return
enable: No help available

get_ident() → List[str]

```
# SCPI: CONFigure:BASE:IPCR:IDENt
value: List[str] = driver.configure.base.ipcr.get_ident()
```

No command help available

return
ident: No help available

set_enable(*enable: List[bool]*) → None

```
# SCPI: CONFigure:BASE:IPCR:ENABle
driver.configure.base.ipcr.set_enable(enable = [True, False, True])
```

No command help available

param enable
No help available

6.6.1.5 IpSet

class IpSetCls

IpSet 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.base.ipSet.clone()
```

Subgroups

6.6.1.5.1 NwAdapter<NwAdapter>

RepCap Settings

```
# Range: Adapter1 .. Adapter5
rc = driver.configure.base.ipSet.nwAdapter.repcap_nwAdapter_get()
driver.configure.base.ipSet.nwAdapter.repcap_nwAdapter_set(repcap.NwAdapter.Adapter1)
```

SCPI Command :

```
CONFigure:BASE:IPSet:NWADapter<n>
```

class NwAdapterCls

NwAdapter commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: NwAdapter, default value after init: NwAdapter.Adapter1

class GetStruct

Response structure. Fields:

- Nw_Adapter_Name: str: No parameter help available
- Set_Subnet_Conform: bool: No parameter help available
- Ip_Address: str: No parameter help available
- Status: enums.AdjustStatus: No parameter help available

get(*nwAdapter=NwAdapter.Default*) → GetStruct

```
# SCPI: CONFigure:BASE:IPSet:NWADapter<n>
value: GetStruct = driver.configure.base.ipSet.nwAdapter.get(nwAdapter = repcap.
↪NwAdapter.Default)
```

No command help available

param nwAdapter
optional repeated capability selector. Default value: Adapter1 (settable in the interface 'NwAdapter')

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

set(*set_subnet_conform: bool*, *nwAdapter=NwAdapter.Default*) → None

```
# SCPI: CONFigure:BASE:IPSet:NWADapter<n>
driver.configure.base.ipSet.nwAdapter.set(set_subnet_conform = False, nwAdapter␣
↪= repcap.NwAdapter.Default)
```

No command help available

param set_subnet_conform
No help available

param nwAdapter
optional repeated capability selector. Default value: Adapter1 (settable in the interface 'NwAdapter')

Cloning the Group

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

6.6.1.6 Mmonitor

class MmonitorCls

Mmonitor 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.base.mmonitor.clone()
```

Subgroups

6.6.1.6.1 IpAddress<IpAddress>

RepCap Settings

```
# Range: Addr1 .. Addr3
rc = driver.configure.base.mmonitor.ipAddress.repcap_ipAddress_get()
driver.configure.base.mmonitor.ipAddress.repcap_ipAddress_set(repcap.IpAddress.Addr1)
```

SCPI Command :

```
CONFigure:BASE:MMONitor:IPADdress<n>
```

class IpAddressCls

IpAddress commands group definition. 1 total commands, 0 Subgroups, 1 group commands Repeated Capability: IpAddress, default value after init: IpAddress.Addr1

class IpAddressStruct

Response structure. Fields:

- First_Segment: int: No parameter help available
- Second_Segment: int: No parameter help available
- System_Id: int: No parameter help available
- Local_Id: int: No parameter help available

get(*ipAddress=IpAddress.Default*) → IpAddressStruct

```
# SCPI: CONFigure:BASE:MMONitor:IPADdress<n>
value: IpAddressStruct = driver.configure.base.mmonitor.ipAddress.get(ipAddress␣
↪= repcap.IpAddress.Default)
```

No command help available

param ipAddress
optional repeated capability selector. Default value: Addr1 (settable in the interface 'IpAddress')

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

set(*first_segment: int*, *second_segment: int*, *system_id: int*, *local_id: int*, *ipAddress=IpAddress.Default*) → None

```
# SCPI: CONFigure:BASE:MMONitor:IPADdress<n>
driver.configure.base.mmonitor.ipAddress.set(first_segment = 1, second_segment␣
↪= 1, system_id = 1, local_id = 1, ipAddress = repcap.IpAddress.Default)
```

No command help available

param first_segment
No help available

param second_segment
No help available

param system_id
No help available

param local_id
No help available

param ipAddress
optional repeated capability selector. Default value: Addr1 (settable in the interface 'IpAddress')

Cloning the Group

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

6.6.1.7 MultiCmw

SCPI Command :

```
CONFigure:BASE:MCMW:REARrange
```

class MultiCmwCls

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

set_rearrange(*box_nr: List[BoxNumber]*) → None

```
# SCPI: CONFigure:BASE:MCMW:REARrange
driver.configure.base.multiCmw.set_rearrange(box_nr = [BoxNumber.BOX1,␣
↪BoxNumber.NAV])
```

No command help available

param box_nr
No help available

Cloning the Group

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

Subgroups

6.6.1.7.1 Identify

SCPI Command :

```
CONFigure:BASE:MCMW:IDENtify:BTIMe
```

class IdentifyCls

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

get_btime() → int

```
# SCPI: CONFigure:BASE:MCMW:IDENtify:BTIMe
value: int = driver.configure.base.multiCmw.identify.get_btime()
```

No command help available

return
blinking_time: No help available

set_btime(*blinking_time: int*) → None

```
# SCPI: CONFigure:BASE:MCMW:IDENtify:BTIMe
driver.configure.base.multiCmw.identify.set_btime(blinking_time = 1)
```

No command help available

param blinking_time
No help available

6.6.1.8 Salignment

SCPI Commands :

```
CONFigure:BASE:SALignment:MODE
CONFigure:BASE:SALignment:SLOT
```

class SalignmentCls

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

get_mode() → SalignmentMode

```
# SCPI: CONFigure:BASE:SALignment:MODE
value: enums.SalignmentMode = driver.configure.base.salignment.get_mode()
```

Selects the measurement mode for self-alignment.

return
mode: Mode IQ, Level, Verify IQ

get_slot() → str

```
# SCPI: CONFigure:BASE:SALignment:SLOT
value: str = driver.configure.base.salignment.get_slot()
```

No command help available

return
slot: No help available

set_mode(*mode: SalignmentMode*) → None

```
# SCPI: CONFigure:BASE:SALignment:MODE
driver.configure.base.salignment.set_mode(mode = enums.SalignmentMode.FLEVel)
```

Selects the measurement mode for self-alignment.

param mode
Mode IQ, Level, Verify IQ

set_slot(*slot: str*) → None

```
# SCPI: CONFigure:BASE:SALignment:SLOT
driver.configure.base.salignment.set_slot(slot = 'abc')
```

No command help available

param slot
No help available

6.6.2 Cmwd

SCPI Command :

```
CONFigure:CMWD:TIMeout
```

class CmwdCls

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

get_timeout() → float

```
# SCPI: CONFigure:CMWD:TIMeout
value: float = driver.configure.cmwd.get_timeout()
```

No command help available

return
timeout: No help available

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:CMWD:TIMeout
driver.configure.cmwd.set_timeout(timeout = 1.0)
```

No command help available

param timeout
No help available

6.6.3 FreqCorrection

SCPI Commands :

```
CONFigure:FDCorrection:DEACtivate
CONFigure:FDCorrection:DEACtivate:ALL
```

class FreqCorrectionCls

FreqCorrection commands group definition. 4 total commands, 2 Subgroups, 2 group commands

deactivate(*connector: str*, *direction: RxTxDirection = None*, *rf_converter: RfConverterInPath = None*) → None

```
# SCPI: CONFigure:FDCorrection:DEACtivate
driver.configure.freqCorrection.deactivate(connector = rawAbc, direction =␣
↪enums.RxTxDirection.RX, rf_converter = enums.RfConverterInPath.RF1)
```

No command help available

param connector
No help available

param direction
No help available

param rf_converter
No help available

deactivate_all(*direction: RxTxDirection = None*, *table_path: str = None*) → None

```
# SCPI: CONFigure:FDCorrection:DEACtivate:ALL
driver.configure.freqCorrection.deactivate_all(direction = enums.RxTxDirection.
↪RX, table_path = rawAbc)
```

No command help available

param direction
No help available

param table_path
No help available

Cloning the Group

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

Subgroups

6.6.3.1 Activate

SCPI Command :

```
CONFigure:FDCorrection:ACTivate
```

class ActivateCls

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

class GetStruct

Response structure. Fields:

- Table_Rx: str: No parameter help available
- Table_Tx: str: No parameter help available

get(*connector: str*) → GetStruct

```
# SCPI: CONFigure:FDCorrection:ACTivate
value: GetStruct = driver.configure.freqCorrection.activate.get(connector =␣
↪rawAbc)
```

No command help available

param connector
No help available

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

set(*connector: str*, *table: str*, *direction: RxTxDirection = None*, *rf_converter: RfConverterInPath = None*) → None

```
# SCPI: CONFigure:FDCorrection:ACTivate
driver.configure.freqCorrection.activate.set(connector = rawAbc, table = 'abc',␣
↪direction = enums.RxTxDirection.RX, rf_converter = enums.RfConverterInPath.
↪RF1)
```

No command help available

param connector
No help available

param table
No help available

param direction
No help available

param rf_converter
No help available

6.6.3.2 Usage

SCPI Command :

```
CONFigure:FDCorrection:USAGe
```

class UsageCls

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

class GetStruct

Response structure. Fields:

- Not_Avail_Rx: str: No parameter help available
- Correction_Table_Rx: str: No parameter help available
- Not_Avail_Tx: str: No parameter help available
- Correction_Table_Tx: str: No parameter help available

get(*connector: str*, *rf_converter: RfConverterInPath = None*) → GetStruct

```
# SCPI: CONFigure:FDCorrection:USAGe
value: GetStruct = driver.configure.freqCorrection.usage.get(connector = rawAbc,
↪ rf_converter = enums.RfConverterInPath.RF1)
```

No command help available

param connector
No help available

param rf_converter
No help available

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

6.6.4 Gprf

class GprfCls

Gprf 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.gprf.clone()
```

Subgroups

6.6.4.1 Measurement

class MeasurementCls

Measurement 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.gprf.measurement.clone()
```

Subgroups

6.6.4.1.1 IqRecorder

class IqRecorderCls

IqRecorder 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.gprf.measurement.iqRecorder.clone()
```

Subgroups

6.6.4.1.1.1 Trigger

SCPI Command :

```
[CONFigure]:GPRF:MEASurement<Instance>:IQRecorder:TRIGger:SOURce
```

class TriggerCls

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

get_source() → TriggerSource

```
# SCPI: [CONFigure]:GPRF:MEASurement<Instance>:IQRecorder:TRIGger:SOURce
value: enums.TriggerSource = driver.configure.gprf.measurement.iqRecorder.
↪trigger.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: TriggerSource*) → None

```
# SCPI: [CONFigure]:GPRF:MEASurement<Instance>:IQRecorder:TRIGger:SOURce
driver.configure.gprf.measurement.iqRecorder.trigger.set_source(trigger = enums.
↪TriggerSource.EXTernal)
```

No command help available

param trigger
No help available

6.6.4.1.2 IqVsSlot

class IqVsSlotCls

IqVsSlot 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.gprf.measurement.iqVsSlot.clone()
```

Subgroups

6.6.4.1.2.1 Trigger

SCPI Command :

```
[CONFigure]:GPRF:MEASurement<Instance>:IQVSlot:TRIGger:SOURce
```

class TriggerCls

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

get_source() → TriggerSource

```
# SCPI: [CONFigure]:GPRF:MEASurement<Instance>:IQVSlot:TRIGger:SOURce
value: enums.TriggerSource = driver.configure.gprf.measurement.iqVsSlot.trigger.
↪get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: TriggerSource*) → None

```
# SCPI: [CONFigure]:GPRF:MEASurement<Instance>:IQVSlot:TRIGger:SOURce
driver.configure.gprf.measurement.iqVsSlot.trigger.set_source(trigger = enums.
↪TriggerSource.EXTernal)
```

No command help available

param trigger
No help available

6.6.4.1.3 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.configure.gprf.measurement.power.clone()
```

Subgroups

6.6.4.1.3.1 Trigger

SCPI Command :

```
[CONFigure]:GPRF:MEASurement<Instance>:POWer:TRIGger:SOURce
```

class TriggerCls

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

get_source() → TriggerSource

```
# SCPI: [CONFigure]:GPRF:MEASurement<Instance>:POWer:TRIGger:SOURce
value: enums.TriggerSource = driver.configure.gprf.measurement.power.trigger.
↪get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: TriggerSource*) → None

```
# SCPI: [CONFigure]:GPRF:MEASurement<Instance>:POWer:TRIGger:SOURce
driver.configure.gprf.measurement.power.trigger.set_source(trigger = enums.
↪TriggerSource.EXTernal)
```

No command help available

param trigger
No help available

6.6.5 Mutex

SCPI Commands :

```
CONFigure:MUTex:UNLock
CONFigure:MUTex:UNDefine
CONFigure:MUTex:CATalog
```

class MutexCls

Mutex commands group definition. 6 total commands, 3 Subgroups, 3 group commands

class CatalogStruct

Structure for reading output parameters. Fields:

- Name: str: No parameter help available
- Def_Timeout: int: No parameter help available
- State: enums.MutexState: No parameter help available

get_catalog() → CatalogStruct

```
# SCPI: CONFigure:MUTex:CATalog
value: CatalogStruct = driver.configure.mutex.get_catalog()
```

No command help available

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

set_undefine(*name: str*) → None

```
# SCPI: CONFigure:MUTex:UNDefine
driver.configure.mutex.set_undefine(name = 'abc')
```

No command help available

param name
No help available

unlock(*name: str*, *key: float*) → None

```
# SCPI: CONFigure:MUTex:UNLock
driver.configure.mutex.unlock(name = 'abc', key = 1.0)
```

No command help available

param name
No help available

param key
No help available

Cloning the Group

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

Subgroups

6.6.5.1 Define

SCPI Command :

```
CONFigure:MUTex:DEFine
```

class DefineCls

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

set(*name: str*, *timeout: float*, *scope: ValidityScopeA = None*) → None

```
# SCPI: CONFigure:MUTex:DEFine
driver.configure.mutex.define.set(name = 'abc', timeout = 1.0, scope = enums.
↪ValidityScopeA.GLOBal)
```

No command help available

param name
No help available

param timeout
No help available

param scope
No help available

6.6.5.2 Lock

SCPI Command :

```
CONFigure:MUTex:LOCK
```

class LockCls

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

get(*name: str*, *timeout: float = None*) → int

```
# SCPI: CONFigure:MUTex:LOCK
value: int = driver.configure.mutex.lock.get(name = 'abc', timeout = 1.0)
```

No command help available

param name
No help available

param timeout
No help available

return
key: No help available

6.6.5.3 State

SCPI Command :

```
CONFigure:MUTex:STATe
```

class StateCls

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

class GetStruct

Response structure. Fields:

- State: enums.MutexState: No parameter help available
- Key: int: No parameter help available

get(*name: str*, *action: MutexAction = None*, *timeout: float = None*) → GetStruct

```
# SCPI: CONFigure:MUTex:STATe
value: GetStruct = driver.configure.mutex.state.get(name = 'abc', action =␣
↪enums.MutexAction.DONothing, timeout = 1.0)
```

No command help available

param name
No help available

param action
No help available

param timeout
No help available

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

6.6.6 Selftest

SCPI Commands :

```
CONFigure:SELFtest:ASMeas
CONFigure:SELFtest:SCONdition
CONFigure:SELFtest:REPetition
CONFigure:SELFtest:SMODe
CONFigure:SELFtest:EXECution
```

class SelftestCls

Selftest commands group definition. 11 total commands, 3 Subgroups, 5 group commands

get_as_meas() → bool

```
# SCPI: CONFigure:SELFtest:ASMeas
value: bool = driver.configure.selftest.get_as_meas()
```

No command help available

return
state: No help available

get_execution() → Execution

```
# SCPI: CONFigure:SELFtest:EXECution
value: enums.Execution = driver.configure.selftest.get_execution()
```

No command help available

return
execution: No help available

get_repetition() → Repeat

```
# SCPI: CONFigure:SELFtest:REPetition
value: enums.Repeat = driver.configure.selftest.get_repetition()
```

No command help available

return
repetition: No help available

get_scondition() → SelftestStopCondition

```
# SCPI: CONFigure:SELFtest:SCONdition
value: enums.SelftestStopCondition = driver.configure.selftest.get_scondition()
```

No command help available

return
stop_condition: No help available

get_smode() → SelftestSpecMode

```
# SCPI: CONFigure:SELFtest:SMODe
value: enums.SelftestSpecMode = driver.configure.selftest.get_smode()
```

No command help available

return
spec_mode: No help available

set_as_meas(*state: bool*) → None

```
# SCPI: CONFigure:SELFtest:ASMeas
driver.configure.selftest.set_as_meas(state = False)
```

No command help available

param state
No help available

set_execution(*execution: Execution*) → None

```
# SCPI: CONFigure:SELFtest:EXECution
driver.configure.selftest.set_execution(execution = enums.Execution.CONCurrent)
```

No command help available

param execution
No help available

set_repetition(*repetition: Repeat*) → None

```
# SCPI: CONFigure:SELFtest:REPetition
driver.configure.selftest.set_repetition(repetition = enums.Repeat.CONTinuous)
```

No command help available

param repetition
No help available

set_scondition(*stop_condition: SelftestStopCondition*) → None

```
# SCPI: CONFigure:SELFtest:SCONdition
driver.configure.selftest.set_scondition(stop_condition = enums.
↪SelftestStopCondition.NONE)
```

No command help available

param stop_condition
No help available

set_smode(*spec_mode: SelftestSpecMode*) → None

```
# SCPI: CONFigure:SELFtest:SMODe
driver.configure.selftest.set_smode(spec_mode = enums.SelftestSpecMode.NONE)
```

No command help available

param spec_mode
No help available

Cloning the Group

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

Subgroups

6.6.6.1 Info

SCPI Command :

```
CONFigure:SELFtest:INFO:PROGress
```

class InfoCls

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

get_progress() → str

```
# SCPI: CONFigure:SELFtest:INFO:PROGress
value: str = driver.configure.selftest.info.get_progress()
```

No command help available

return
progress: No help available

Cloning the Group

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

Subgroups

6.6.6.1.1 Description

SCPI Command :

```
CONFigure:SELFtest:INFO:DESCription
```

class DescriptionCls

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

get(*test_name: str*) → str

```
# SCPI: CONFigure:SELFtest:INFO:DESCription
value: str = driver.configure.selftest.info.description.get(test_name = 'abc')
```

No command help available

param test_name
No help available

return
detailed_desc: No help available

6.6.6.1.2 Message

SCPI Command :

```
CONFigure:SELFtest:INFO:MESSage
```

class MessageCls

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

get(*test_name: str*) → str

```
# SCPI: CONFigure:SELFtest:INFO:MESSage
value: str = driver.configure.selftest.info.message.get(test_name = 'abc')
```

No command help available

param test_name
No help available

return
message: No help available

6.6.6.2 Select

SCPI Command :

```
CONFigure:SELFtest:SELect
```

class SelectCls

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

set(*state: bool*, *filter_py: str = None*) → None

```
# SCPI: CONFigure:SELFtest:SELect
driver.configure.selftest.select.set(state = False, filter_py = 'abc')
```

No command help available

param state
No help available

param filter_py
No help available

6.6.6.3 Uprofile

SCPI Commands :

```
CONFigure:SELFtest:UPRofile:SAVE
CONFigure:SELFtest:UPRofile:LOAD
```

class UprofileCls

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

get_load() → str

```
# SCPI: CONFigure:SELFtest:UPRofile:LOAD
value: str = driver.configure.selftest.uprofile.get_load()
```

No command help available

return
user_profile: No help available

save(*save_user_profile: str*) → None

```
# SCPI: CONFigure:SELFtest:UPRofile:SAVE
driver.configure.selftest.uprofile.save(save_user_profile = 'abc')
```

No command help available

param save_user_profile
No help available

set_load(*user_profile: str*) → None

```
# SCPI: CONFigure:SELFtest:UPRofile:LOAD
driver.configure.selftest.uprofile.set_load(user_profile = 'abc')
```

No command help available

param user_profile
No help available

6.6.7 Semaphore

SCPI Commands :

```
CONFigure:SEMaphore:CATalog
CONFigure:SEMaphore:UNDefine
```

class SemaphoreCls

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

class CatalogStruct

Structure for reading output parameters. Fields:

- Name: str: No parameter help available
- Def_Timeout: int: No parameter help available
- Def_Count: int: No parameter help available
- Scope: enums.ValidityScopeA: No parameter help available

get_catalog() → CatalogStruct

```
# SCPI: CONFigure:SEMaphore:CATalog
value: CatalogStruct = driver.configure.semaphore.get_catalog()
```

No command help available

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

set_undefine(*name: str*) → None

```
# SCPI: CONFigure:SEMaphore:UNDefine
driver.configure.semaphore.set_undefine(name = 'abc')
```

No command help available

param name
No help available

Cloning the Group

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

Subgroups

6.6.7.1 Acquire

SCPI Command :

```
CONFigure:SEMaphore:ACQuire
```

class AcquireCls

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

get(*name: str*) → int

```
# SCPI: CONFigure:SEMaphore:ACQuire
value: int = driver.configure.semaphore.acquire.get(name = 'abc')
```

No command help available

param name
No help available

return
key: No help available

6.6.7.2 Count

SCPI Command :

```
CONFigure:SEMaphore:COUNt
```

class CountCls

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

get(*name: str*) → int

```
# SCPI: CONFigure:SEMaphore:COUNt
value: int = driver.configure.semaphore.count.get(name = 'abc')
```

No command help available

param name
No help available

return
count: No help available

6.6.7.3 Define

SCPI Command :

```
CONFigure:SEMaphore:DEFine
```

class DefineCls

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

set(*name: str*, *def_timeout: float*, *def_count: int*, *scope: ValidityScopeA = None*) → None

```
# SCPI: CONFigure:SEMaphore:DEFine
driver.configure.semaphore.define.set(name = 'abc', def_timeout = 1.0, def_
↪count = 1, scope = enums.ValidityScopeA.GLOBal)
```

No command help available

param name
No help available

param def_timeout
No help available

param def_count
No help available

param scope
No help available

6.6.7.4 Release

SCPI Command :

```
CONFigure:SEMaphore:RELease
```

class ReleaseCls

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

set(*name: str*, *key: int*) → None

```
# SCPI: CONFigure:SEMaphore:RELease
driver.configure.semaphore.release.set(name = 'abc', key = 1)
```

No command help available

param name
No help available

param key
No help available

6.6.8 SingleCmw

class SingleCmwCls

SingleCmw commands group definition. 8 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.6.8.1 FreqCorrection

SCPI Command :

```
CONFigure:CMWS:FDCorrection:DEACtivate:ALL
```

class FreqCorrectionCls

FreqCorrection commands group definition. 8 total commands, 3 Subgroups, 1 group commands

deactivate_all(*table_path: str = None*) → None

```
# SCPI: CONFigure:CMWS:FDCorrection:DEACtivate:ALL
driver.configure.singleCmw.freqCorrection.deactivate_all(table_path = rawAbc)
```

No command help available

param table_path
No help available

Cloning the Group

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

Subgroups

6.6.8.1.1 Activate

class ActivateCls

Activate 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.singleCmw.freqCorrection.activate.clone()
```

Subgroups

6.6.8.1.1.1 Rx

SCPI Command :

```
CONFigure:CMWS:FDCorrection:ACTivate:RX
```

class RxCls

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

class SetStruct

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

- Connector_Bench: str: No parameter help available
- Table_1: str: No parameter help available
- Table_2: str: No parameter help available
- Table_3: str: No parameter help available
- Table_4: str: No parameter help available
- Table_5: str: No parameter help available
- Table_6: str: No parameter help available
- Table_7: str: No parameter help available
- Table_8: str: No parameter help available

set(*structure: SetStruct*) → None

```
# SCPI: CONFigure:CMWS:FDCorrection:ACTivate:RX
structure = driver.configure.singleCmw.freqCorrection.activate.rx.SetStruct()
structure.Connector_Bench: str = rawAbc
structure.Table_1: str = 'abc'
structure.Table_2: str = 'abc'
structure.Table_3: str = 'abc'
structure.Table_4: str = 'abc'
structure.Table_5: str = 'abc'
structure.Table_6: str = 'abc'
```

(continues on next page)

(continued from previous page)

```
structure.Table_7: str = 'abc'
structure.Table_8: str = 'abc'
driver.configure.singleCmw.freqCorrection.activate.rx.set(structure)
```

No command help available

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

6.6.8.1.1.2 Tx

SCPI Command :

```
CONFigure:CMWS:FDCorrection:ACTivate:TX
```

class TxCls

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

class SetStruct

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

- Connector_Bench: str: No parameter help available
- Table_1: str: No parameter help available
- Table_2: str: No parameter help available
- Table_3: str: No parameter help available
- Table_4: str: No parameter help available
- Table_5: str: No parameter help available
- Table_6: str: No parameter help available
- Table_7: str: No parameter help available
- Table_8: str: No parameter help available

set(*structure: SetStruct*) → None

```
# SCPI: CONFigure:CMWS:FDCorrection:ACTivate:TX
structure = driver.configure.singleCmw.freqCorrection.activate.tx.SetStruct()
structure.Connector_Bench: str = rawAbc
structure.Table_1: str = 'abc'
structure.Table_2: str = 'abc'
structure.Table_3: str = 'abc'
structure.Table_4: str = 'abc'
structure.Table_5: str = 'abc'
structure.Table_6: str = 'abc'
structure.Table_7: str = 'abc'
structure.Table_8: str = 'abc'
driver.configure.singleCmw.freqCorrection.activate.tx.set(structure)
```

No command help available

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

6.6.8.1.2 Deactivate

class DeactivateCls

Deactivate 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.singleCmw.freqCorrection.deactivate.clone()
```

Subgroups

6.6.8.1.2.1 Rx

SCPI Command :

```
CONFigure:CMWS:FDCorrection:DEACtivate:RX
```

class RxCls

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

set_value(*connector_bench: str*) → None

```
# SCPI: CONFigure:CMWS:FDCorrection:DEACtivate:RX
driver.configure.singleCmw.freqCorrection.deactivate.rx.set_value(connector_
↪bench = rawAbc)
```

No command help available

param connector_bench
No help available

Cloning the Group

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

Subgroups

6.6.8.1.2.2 All

SCPI Command :

```
CONFigure:CMWS:FDCorrection:DEACtivate:RX:ALL
```

class AllCls

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

set(*table_path: str = None*) → None

```
# SCPI: CONFigure:CMWS:FDCorrection:DEACtivate:RX:ALL
driver.configure.singleCmw.freqCorrection.deactivate.rx.all.set(table_path =␣
↪rawAbc)
```

No command help available

param table_path
No help available

6.6.8.1.2.3 Tx

SCPI Command :

```
CONFigure:CMWS:FDCorrection:DEACtivate:TX
```

class TxCls

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

set_value(*connector_bench: str*) → None

```
# SCPI: CONFigure:CMWS:FDCorrection:DEACtivate:TX
driver.configure.singleCmw.freqCorrection.deactivate.tx.set_value(connector_
↪bench = rawAbc)
```

No command help available

param connector_bench
No help available

Cloning the Group

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

Subgroups

6.6.8.1.2.4 All

SCPI Command :

```
CONFigure:CMWS:FDCorrection:DEACtivate:TX:ALL
```

class AllCls

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

set(*table_path: str = None*) → None

```
# SCPI: CONFigure:CMWS:FDCorrection:DEACtivate:TX:ALL
driver.configure.singleCmw.freqCorrection.deactivate.tx.all.set(table_path =␣
↪rawAbc)
```

No command help available

param table_path
No help available

6.6.8.1.3 Usage

SCPI Command :

```
CONFigure:CMWS:FDCorrection:USAGe
```

class UsageCls

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

class GetStruct

Response structure. Fields:

- Correction_Table_Rx: str: No parameter help available
- Correction_Table_Tx: str: No parameter help available

get(*connector: str*) → GetStruct

```
# SCPI: CONFigure:CMWS:FDCorrection:USAGe
value: GetStruct = driver.configure.singleCmw.freqCorrection.usage.
↪get(connector = rawAbc)
```

No command help available

param connector
No help available

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

6.6.9 Spoint

SCPI Commands :

```
CONFigure:SPOint:CATalog
CONFigure:SPOint:UNDefine
```

class SpointCls

Spoint commands group definition. 5 total commands, 3 Subgroups, 2 group commands

class CatalogStruct

Structure for reading output parameters. Fields:

- Name: str: No parameter help available
- Def_Timeout: int: No parameter help available

- Def_Count: int: No parameter help available
- Scope: enums.ValidityScopeA: No parameter help available

get_catalog() → CatalogStruct

```
# SCPI: CONFigure:SPOint:CATalog
value: CatalogStruct = driver.configure.spoint.get_catalog()
```

No command help available

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

set_undefine(*name: str*) → None

```
# SCPI: CONFigure:SPOint:UNDefine
driver.configure.spoint.set_undefine(name = 'abc')
```

No command help available

param name
No help available

Cloning the Group

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

Subgroups

6.6.9.1 Define

SCPI Command :

```
CONFigure:SPOint:DEFine
```

class DefineCls

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

set(*name: str*, *def_timeout: float*, *def_count: int*, *scope: ValidityScopeA = None*) → None

```
# SCPI: CONFigure:SPOint:DEFine
driver.configure.spoint.define.set(name = 'abc', def_timeout = 1.0, def_count =␣
↪1, scope = enums.ValidityScopeA.GLOBal)
```

No command help available

param name
No help available

param def_timeout
No help available

param def_count
No help available

param scope
No help available

6.6.9.2 Join

SCPI Command :

```
CONFigure:SPOint:JOIN
```

class JoinCls

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

class GetStruct

Response structure. Fields:

- Count: int: No parameter help available
- Result: enums.SyncResult: No parameter help available

get(*name: str*, *action: JoinAction = None*, *polling: SyncPolling = None*, *poll_interval: float = None*) → GetStruct

```
# SCPI: CONFigure:SPOint:JOIN
value: GetStruct = driver.configure.spoint.join.get(name = 'abc', action =␣
↪enums.JoinAction.CTASk, polling = enums.SyncPolling.NPOLling, poll_interval =␣
↪1.0)
```

No command help available

param name
No help available

param action
No help available

param polling
No help available

param poll_interval
No help available

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

6.6.9.3 Rewait

SCPI Command :

```
CONFigure:SPOint:REWait
```

class RewaitCls

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

class GetStruct

Response structure. Fields:

- Count: int: No parameter help available
- Result: enums.SyncResult: No parameter help available

get(*name: str*) → GetStruct

```
# SCPI: CONFigure:SPOint:REWait
value: GetStruct = driver.configure.spoint.rewait.get(name = 'abc')
```

No command help available

param name
No help available

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

6.6.10 System

class SystemCls

System commands group definition. 17 total commands, 11 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.6.10.1 Attenuation

class AttenuationCls

Attenuation 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.system.attenuation.clone()
```

Subgroups

6.6.10.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.system.attenuation.correctionTable.clone()
```

Subgroups

6.6.10.1.1.1 Info

class InfoCls

Info 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.system.attenuation.correctionTable.info.clone()
```

Subgroups

6.6.10.1.1.2 Globale

SCPI Command :

```
[CONFigure]:SYSTem:ATTenuation:CTABle:INFO:GLOBal
```

class GlobaleCls

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

class GetStruct

Response structure. Fields:

- Frequency: List[float]: No parameter help available
- Attenuation: List[float]: No parameter help available

get(*name: str*) → GetStruct

```
# SCPI: [CONFigure]:SYSTem:ATTenuation:CTABle:INFO:GLOBal
value: GetStruct = driver.configure.system.attenuation.correctionTable.info.
↪globale.get(name = 'abc')
```

Returns the entries of a global correction table as pairs of values: {<Frequency>, <Attenuation>}1, {<Frequency>, <Attenuation>}2, ...

param name
Name of the correction table.

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

6.6.10.1.1.3 Tenvironment

SCPI Command :

```
[CONFigure]:SYSTem:ATTenuation:CTABle:INFO[:TENVironment]
```

class TenvironmentCls

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

class GetStruct

Response structure. Fields:

- Frequency: List[float]: No parameter help available
- Attenuation: List[float]: No parameter help available

get(*name: str*) → GetStruct

```
# SCPI: [CONFigure]:SYSTem:ATTenuation:CTABle:INFO[:TENVironment]
value: GetStruct = driver.configure.system.attenuation.correctionTable.info.
↪tenvironment.get(name = 'abc')
```

Returns the entries of a channel-specific correction table as pairs of values: {<Frequency>, <Attenuation>}1, {<Frequency>, <Attenuation>}2, ...

param name
Name of the correction table.

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

6.6.10.2 Control

class ControlCls

Control 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.system.control.clone()
```

Subgroups

6.6.10.2.1 Reboot

SCPI Command :

```
[CONFigure]:SYSTem:CONTrol:REBoot
```

class RebootCls

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

set() → None

```
# SCPI: [CONFigure]:SYSTem:CONTrol:REBoot
driver.configure.system.control.reboot.set()
```

Reboots the instrument / its operating system.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: [CONFigure]:SYSTem:CONTrol:REBoot
driver.configure.system.control.reboot.set_with_opc()
```

Reboots the instrument / its operating system.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.6.10.2.2 Restart

SCPI Command :

```
[CONFigure]:SYSTem:CONTrol:RESTart
```

class RestartCls

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

set() → None

```
# SCPI: [CONFigure]:SYSTem:CONTrol:RESTart
driver.configure.system.control.restart.set()
```

Restarts the test software on the instrument.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: [CONFigure]:SYSTem:CONTrol:RESTart
driver.configure.system.control.restart.set_with_opc()
```

Restarts the test software on the instrument.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.6.10.2.3 Shutdown

SCPI Command :

```
[CONFigure]:SYSTem:CONTrol:SHUTdown
```

class ShutdownCls

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

set() → None

```
# SCPI: [CONFigure]:SYSTem:CONTrol:SHUTdown
driver.configure.system.control.shutdown.set()
```

Brings the instrument into the standby state.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: [CONFigure]:SYSTem:CONTrol:SHUTdown
driver.configure.system.control.shutdown.set_with_opc()
```

Brings the instrument into the standby state.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.6.10.3 Edevice

SCPI Command :

```
[CONFigure]:SYSTem:EDEVice
```

class EdeviceCls

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

class EdeviceStruct

Response structure. Fields:

- Device_Type: enums.DeviceType: No parameter help available
- Device_Mode: enums.DeviceMode: No parameter help available

get() → EdeviceStruct

```
# SCPI: [CONFigure]:SYSTem:EDEVice
value: EdeviceStruct = driver.configure.system.edevice.get()
```

No command help available

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

set(*device_type: DeviceType*, *device_mode: DeviceMode*) → None

```
# SCPI: [CONFigure]:SYSTem:EDEVice
driver.configure.system.edevice.set(device_type = enums.DeviceType.NONE, device_
↪mode = enums.DeviceMode.M2X2)
```

No command help available

param device_type
No help available

param device_mode
No help available

6.6.10.4 Recall

class RecallCls

Recall 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.system.recall.clone()
```

Subgroups

6.6.10.4.1 Partial

SCPI Command :

```
[CONFigure]:SYSTem:RECall:PARTial
```

class PartialCls

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

get(*saving_path: str*) → List[str]

```
# SCPI: [CONFigure]:SYSTem:RECall:PARTial
value: List[str] = driver.configure.system.recall.partial.get(saving_path = 'abc
↪')
```

Recalls the instrument settings from a file. A query returns a comma-separated list of all components contained in the file.

param saving_path
No help available

return
saving_module: No help available

set(*saving_path: str*, *saving_module: List[str] = None*) → None

```
# SCPI: [CONFigure]:SYSTem:RECall:PARTial
driver.configure.system.recall.partial.set(saving_path = 'abc', saving_module =␣
↪['abc1', 'abc2', 'abc3'])
```

Recalls the instrument settings from a file. A query returns a comma-separated list of all components contained in the file.

param saving_path
No help available

param saving_module
No help available

6.6.10.5 Reset

SCPI Command :

```
[CONFigure]:SYSTem:RESet:PARTial
```

class ResetCls

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

set_partial(*resetable_system_part: List[str]*) → None

```
# SCPI: [CONFigure]:SYSTem:RESet:PARTial
driver.configure.system.reset.set_partial(resetable_system_part = ['abc1', 'abc2
↪', 'abc3'])
```

Resets selected system components.

param resetable_system_part
Comma-separated list of strings, one string per component to be reset.

6.6.10.6 Rf42

class Rf42Cls

Rf42 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.system.rf42.clone()
```

Subgroups

6.6.10.6.1 Box<Box>

RepCap Settings

```
# Range: Nr1 .. Nr32
rc = driver.configure.system.rf42.box.repcap_box_get()
driver.configure.system.rf42.box.repcap_box_set(repcap.Box.Nr1)
```

class BoxCls

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.system.rf42.box.clone()
```

Subgroups

6.6.10.6.1.1 Apreset

class ApresetCls

Apreset 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.system.rf42.box.apreset.clone()
```

Subgroups

6.6.10.6.1.2 Rx

SCPI Command :

```
[CONFigure]:SYSTem:RF42:BOX<BoxNo>:APReset:RX
```

class RxCls

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

get(*box=Box.Default*) → Amplification

```
# SCPI: [CONFigure]:SYSTem:RF42:BOX<BoxNo>:APReset:RX
value: enums.Amplification = driver.configure.system.rf42.box.apreset.rx.
↪get(box = repcap.Box.Default)
```

Configures the internal amplifier of the R&S CMX-RF42 for the uplink direction.

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

return
amplification: No help available

set(*amplification: Amplification*, *box=Box.Default*) → None

```
# SCPI: [CONFigure]:SYSTem:RF42:BOX<BoxNo>:APReset:RX
driver.configure.system.rf42.box.apreset.rx.set(amplification = enums.
↪Amplification.HIGH, box = repcap.Box.Default)
```

Configures the internal amplifier of the R&S CMX-RF42 for the uplink direction.

param amplification
No help available

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

6.6.10.6.1.3 Tx

SCPI Command :

```
[CONFigure]:SYSTem:RF42:BOX<BoxNo>:APReset:TX
```

class TxCls

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

get(*box=Box.Default*) → LowHigh

```
# SCPI: [CONFigure]:SYSTem:RF42:BOX<BoxNo>:APReset:TX
value: enums.LowHigh = driver.configure.system.rf42.box.apreset.tx.get(box =␣
↪repcap.Box.Default)
```

Configures the internal amplifier of the R&S CMX-RF42 for the downlink direction.

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

return
amplification: No help available

set(*amplification: LowHigh*, *box=Box.Default*) → None

```
# SCPI: [CONFigure]:SYSTem:RF42:BOX<BoxNo>:APReset:TX
driver.configure.system.rf42.box.apreset.tx.set(amplification = enums.LowHigh.
↪HIGH, box = repcap.Box.Default)
```

Configures the internal amplifier of the R&S CMX-RF42 for the downlink direction.

param amplification
No help available

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

6.6.10.7 Rrhead

class RrheadCls

Rrhead 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.system.rrhead.clone()
```

Subgroups

6.6.10.7.1 Lo

class LoCls

Lo 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.system.rrhead.lo.clone()
```

Subgroups

6.6.10.7.1.1 Source

class SourceCls

Source 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.system.rrhead.lo.source.clone()
```

Subgroups

6.6.10.7.1.2 Rx

SCPI Command :

```
CONFigure:SYSTem:RRHead:LO:SOURce:RX
```

class RxCls

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

get(*rrh_name: str*) → SourceInt

```
# SCPI: CONFigure:SYSTem:RRHead:LO:SOURce:RX
value: enums.SourceInt = driver.configure.system.rrhead.lo.source.rx.get(rrh_
↪name = 'abc')
```

Selects the source of the LO signal for the RX direction of the RRH with the <RRH_Name>.

param rrh_name
No help available

return
source: No help available

set(*rrh_name: str*, *source: SourceInt*) → None

```
# SCPI: CONFigure:SYSTem:RRHead:LO:SOURce:RX
driver.configure.system.rrhead.lo.source.rx.set(rrh_name = 'abc', source =␣
↪enums.SourceInt.EXTernal)
```

Selects the source of the LO signal for the RX direction of the RRH with the <RRH_Name>.

param rrh_name
No help available

param source
No help available

6.6.10.7.1.3 Tx

SCPI Command :

```
CONFigure:SYSTem:RRHead:LO:SOURce:TX
```

class TxCls

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

get(*rrh_name: str*) → SourceInt

```
# SCPI: CONFigure:SYSTem:RRHead:LO:SOURce:TX
value: enums.SourceInt = driver.configure.system.rrhead.lo.source.tx.get(rrh_
↪name = 'abc')
```

Selects the source of the LO signal for the TX direction of the RRH with the <RRH_Name>.

param rrh_name
No help available

return
source: No help available

set(*rrh_name: str*, *source: SourceInt*) → None

```
# SCPI: CONFigure:SYSTem:RRHead:LO:SOURce:TX
driver.configure.system.rrhead.lo.source.tx.set(rrh_name = 'abc', source =␣
↪enums.SourceInt.EXTernal)
```

Selects the source of the LO signal for the TX direction of the RRH with the <RRH_Name>.

param rrh_name
No help available

param source
No help available

6.6.10.8 Save

class SaveCls

Save 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.system.save.clone()
```

Subgroups

6.6.10.8.1 Partial

SCPI Command :

```
[CONFigure]:SYSTem:SAVE:PARTial
```

class PartialCls

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

set(*saving_path: str*, *saving_module: List[str]*) → None

```
# SCPI: [CONFigure]:SYSTem:SAVE:PARTial
driver.configure.system.save.partial.set(saving_path = 'abc', saving_module = [
↪'abc1', 'abc2', 'abc3'])
```

Saves the instrument settings to a file. The names of the components are the same as for a partial reset. So you can query them via method RsCMPX_Base.Catalog.System.Reset.partial.

param saving_path
No help available

param saving_module
No help available

6.6.10.9 Vse

SCPI Commands :

```
[CONFigure]:SYSTem:VSE:CONNect
[CONFigure]:SYSTem:VSE:DISConnect
```

class VseCls

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

get_connect() → str

```
# SCPI: [CONFigure]:SYSTem:VSE:CONNect
value: str = driver.configure.system.vse.get_connect()
```

Establishes a connection to the R&S VSE software at the specified <Address>.

return
address: IP address or host name

get_disconnect() → str

```
# SCPI: [CONFigure]:SYSTem:VSE:DISConnect
value: str = driver.configure.system.vse.get_disconnect()
```

Terminates a connection to the R&S VSE software at the specified <Address>.

return
address: IP address or host name

set_connect(*address: str*) → None

```
# SCPI: [CONFigure]:SYSTem:VSE:CONNect
driver.configure.system.vse.set_connect(address = 'abc')
```

Establishes a connection to the R&S VSE software at the specified <Address>.

param address
IP address or host name

set_disconnect(*address: str*) → None

```
# SCPI: [CONFigure]:SYSTem:VSE:DISConnect
driver.configure.system.vse.set_disconnect(address = 'abc')
```

Terminates a connection to the R&S VSE software at the specified <Address>.

param address
IP address or host name

6.6.10.10 Z310

class Z310Cls

Z310 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.system.z310.clone()
```

Subgroups

6.6.10.10.1 Attenuation

SCPI Command :

```
[CONFigure]:SYSTem:Z310:ATTenuation
```

class AttenuationCls

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

get(*connector_name: str*) → float

```
# SCPI: [CONFigure]:SYSTem:Z310:ATTenuation
value: float = driver.configure.system.z310.attenuation.get(connector_name =
↪'abc')
```

No command help available

param connector_name
No help available

return
attenuation: No help available

set(*attenuation: float*) → None

```
# SCPI: [CONFigure]:SYSTem:Z310:ATTenuation
driver.configure.system.z310.attenuation.set(attenuation = 1.0)
```

No command help available

param attenuation
No help available

6.6.10.11 Z320

class Z320Cls

Z320 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.system.z320.clone()
```

Subgroups

6.6.10.11.1 Attenuation

SCPI Command :

```
[CONFigure]:SYSTem:Z320:ATTenuation
```

class AttenuationCls

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

get(*connector_name: str*) → float

```
# SCPI: [CONFigure]:SYSTem:Z320:ATTenuation
value: float = driver.configure.system.z320.attenuation.get(connector_name =
↪'abc')
```

No command help available

param connector_name
No help available

return
attenuation: No help available

set(*attenuation: float*) → None

```
# SCPI: [CONFigure]:SYSTem:Z320:ATTenuation
driver.configure.system.z320.attenuation.set(attenuation = 1.0)
```

No command help available

param attenuation
No help available

6.6.11 Tenvironment

class TenvironmentCls

Tenvironment commands group definition. 6 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.6.11.1 Spath

class SpathCls

Spath 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.configure.tenvironment.spath.clone()
```

Subgroups

6.6.11.1.1 Attenuation

class AttenuationCls

Attenuation 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.tenvironment.spath.attenuation.clone()
```

Subgroups

6.6.11.1.1.1 Rx

SCPI Command :

```
[CONFigure]:TENVironment:SPATh:ATTenuation:RX
```

class RxCls

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

get(*name_signal_path: str*) → float

```
# SCPI: [CONFigure]:TENVironment:SPATh:ATTenuation:RX
value: float = driver.configure.tenvironment.spath.attenuation.rx.get(name_
↪signal_path = 'abc')
```

Assigns a frequency-independent correction value to the TX direction or RX direction of a connection. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

return
value: Attenuation

set(*name_signal_path: str*, *value: float*) → None

```
# SCPI: [CONFigure]:TENVironment:SPATh:ATTenuation:RX
driver.configure.tenvironment.spath.attenuation.rx.set(name_signal_path = 'abc',
↪ value = 1.0)
```

Assigns a frequency-independent correction value to the TX direction or RX direction of a connection. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

param value
Attenuation

6.6.11.1.1.2 Tx

SCPI Command :

```
[CONFigure]:TENVironment:SPATh:ATTenuation:TX
```

class TxCls

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

get(*name_signal_path: str*) → float

```
# SCPI: [CONFigure]:TENVironment:SPATh:ATTenuation:TX
value: float = driver.configure.tenvironment.spath.attenuation.tx.get(name_
↪signal_path = 'abc')
```

Assigns a frequency-independent correction value to the TX direction or RX direction of a connection. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

return
value: Attenuation

set(*name_signal_path: str*, *value: float*) → None

```
# SCPI: [CONFigure]:TENVironment:SPATh:ATTenuation:TX
driver.configure.tenvironment.spath.attenuation.tx.set(name_signal_path = 'abc',
↪ value = 1.0)
```

Assigns a frequency-independent correction value to the TX direction or RX direction of a connection. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

param value
Attenuation

6.6.11.1.2 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.tenvironment.spath.correctionTable.clone()
```

Subgroups

6.6.11.1.2.1 Rx

SCPI Command :

```
[CONFigure]:TENVironment:SPATh:CTABle:RX
```

class RxCls

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

get(*name_signal_path: str*) → List[str]

```
# SCPI: [CONFigure]:TENVironment:SPATh:CTABle:RX
value: List[str] = driver.configure.tenvironment.spath.correctionTable.rx.
↪get(name_signal_path = 'abc')
```

Assigns one or more correction tables to the TX direction or RX direction of a connection. If there is an existing assignment, it is overwritten. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

return
correction_table: The name of the correction table to be assigned. At least one name of a correction table. To assign several tables, use a comma-separated list of strings.

set(*name_signal_path: str*, *correction_table: List[str]*) → None

```
# SCPI: [CONFigure]:TENVironment:SPATh:CTABle:RX
driver.configure.tenvironment.spath.correctionTable.rx.set(name_signal_path =
↪'abc', correction_table = ['abc1', 'abc2', 'abc3'])
```

Assigns one or more correction tables to the TX direction or RX direction of a connection. If there is an existing assignment, it is overwritten. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

param correction_table
The name of the correction table to be assigned. At least one name of a correction table. To assign several tables, use a comma-separated list of strings.

6.6.11.1.2.2 Tx

SCPI Command :

```
[CONFigure]:TENVironment:SPATh:CTABle:TX
```

class TxCls

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

get(*name_signal_path: str*) → List[str]

```
# SCPI: [CONFigure]:TENVironment:SPATh:CTABle:TX
value: List[str] = driver.configure.tenvironment.spath.correctionTable.tx.
↪get(name_signal_path = 'abc')
```

Assigns one or more correction tables to the TX direction or RX direction of a connection. If there is an existing assignment, it is overwritten. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

return
correction_table: The name of the correction table to be assigned. At least one name of a correction table. To assign several tables, use a comma-separated list of strings.

set(*name_signal_path: str*, *correction_table: List[str]*) → None

```
# SCPI: [CONFigure]:TENVironment:SPATh:CTABle:TX
driver.configure.tenvironment.spath.correctionTable.tx.set(name_signal_path =
↪'abc', correction_table = ['abc1', 'abc2', 'abc3'])
```

Assigns one or more correction tables to the TX direction or RX direction of a connection. If there is an existing assignment, it is overwritten. The directions refer to the instrument (TX/RX of the instrument) .

param name_signal_path
Name of the connection

param correction_table
The name of the correction table to be assigned. At least one name of a correction table. To assign several tables, use a comma-separated list of strings.

6.6.11.1.3 Direction

SCPI Command :

```
[CONFigure]:TENVironment:SPATh:DIRection
```

class DirectionCls

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

get(*name_signal_path: str*) → RxTxDirection

```
# SCPI: [CONFigure]:TENVironment:SPATh:DIRection
value: enums.RxTxDirection = driver.configure.tenvironment.spath.direction.
↪get(name_signal_path = 'abc')
```

Defines the direction of a connection (TX/RX of the instrument) .

param name_signal_path
No help available

return
signal_direction: No help available

set(*name_signal_path: str*, *signal_direction: RxTxDirection*) → None

```
# SCPI: [CONFigure]:TENVironment:SPATh:DIRection
driver.configure.tenvironment.spath.direction.set(name_signal_path = 'abc',␣
↪signal_direction = enums.RxTxDirection.RX)
```

Defines the direction of a connection (TX/RX of the instrument) .

param name_signal_path
No help available

param signal_direction
No help available

6.6.11.1.4 Info

SCPI Command :

```
[CONFigure]:TENVironment:SPATh:INFO
```

class InfoCls

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

class GetStruct

Response structure. Fields:

- Name_Antenna: str: Name of the DUT antenna connector.
- Name_Connector: str: Name of the instrument connector.
- Signal_Direction: enums.RxTxDirection: Signal direction, from the point of view of the instrument.
- No_Corr_Table_Rx: float: Number of correction tables assigned to the RX direction of the connection.

- Corr_Table_Rx: str: Comma-separated list of NoCorrTableRX strings. Each string indicates the name of a correction table assigned to the RX direction.
- No_Corr_Table_Tx: float: Number of correction tables assigned to the TX direction of the connection.
- Corr_Table_Tx: List[str]: Comma-separated list of NoCorrTableTX strings. Each string indicates the name of a correction table assigned to the TX direction.

get(*name_spath: str*) → GetStruct

```
# SCPI: [CONFigure]:TENVironment:SPATh:INFO
value: GetStruct = driver.configure.tenvironment.spath.info.get(name_spath =
↪'abc')
```

Returns information about the connection <NameSpath>.

param name_spath
No help available

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

6.7 Create

class CreateCls

Create 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.create.clone()
```

Subgroups

6.7.1 System

class SystemCls

System 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.create.system.clone()
```

Subgroups

6.7.1.1 Attenuation

class AttenuationCls

Attenuation 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.create.system.attenuation.clone()
```

Subgroups

6.7.1.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.create.system.attenuation.correctionTable.clone()
```

Subgroups

6.7.1.1.1.1 Globale

SCPI Command :

```
CREate:SYSTem:ATTenuation:CTABle:GLOBal
```

class GlobaleCls

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

set(*name: str*, *arg_1: List[float] = None*, *attenuation: List[float] = None*) → None

```
# SCPI: CREate:SYSTem:ATTenuation:CTABle:GLOBal
driver.create.system.attenuation.correctionTable.globale.set(name = 'abc', arg_
↪1 = [1.1, 2.2, 3.3], attenuation = [1.1, 2.2, 3.3])
```

Creates a global correction table. You can specify one or more parameter pairs <Frequency>, <Attenuation> to add entries to the table.

param name

Name of the table. Freely configurable and used in other commands to address the table. If a global table with the given name exists already, this table is overwritten.

param arg_1
No help available

param attenuation
No help available

6.7.1.1.1.2 Tenvironment

SCPI Command :

```
CREate:SYSTem:ATTenuation:CTABle[:TENVironment]
```

class TenvironmentCls

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

set(*name: str*, *frequency: List[float] = None*, *attenuation: List[float] = None*) → None

```
# SCPI: CREate:SYSTem:ATTenuation:CTABle[:TENVironment]
driver.create.system.attenuation.correctionTable.tenvironment.set(name = 'abc',␣
↪frequency = [1.1, 2.2, 3.3], attenuation = [1.1, 2.2, 3.3])
```

Creates a channel-specific correction table. You can specify one or more parameter pairs <Frequency>, <Attenuation> to add entries to the table.

param name
Name of the table. Freely configurable and used in other commands to address the table. If a table with the given name exists already for the addressed smart channel, this table is overwritten.

param frequency
No help available

param attenuation
No help available

6.7.2 Tenvironment

class TenvironmentCls

Tenvironment 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.tenvironment.clone()
```

Subgroups

6.7.2.1 Spath

SCPI Command :

```
CREate:TENVironment:SPATh
```

class SpathCls

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

set(*name_signal_path: str*, *name_antenna: str*, *name_connector: str*, *overwrite: bool = None*) → None

```
# SCPI: CREate:TENVironment:SPATh
driver.create.tenvironment.spath.set(name_signal_path = 'abc', name_antenna =
↪'abc', name_connector = 'abc', overwrite = False)
```

Creates a connection for a selected connector and assigns a name to the connection. 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_signal_path
Name of the connection. Freely configurable and used in other commands to address the connection. If a connection with this name already exists, the behavior depends on Overwrite.

param name_antenna
Name of the DUT antenna connector.

param name_connector
Name of the instrument connector. Examples: '0.Slot1.Port1.RRH.RF1', '0.Slot1.Port1.RRH.RF2', '0.Slot1.Port2.IFIn', '0.Slot1.Port2.IFOut'

param overwrite
Selects the behavior if a connection with the NameSignalPath already exists. OFF | 0: No overwrite, return an error. ON | 1: Overwrite the existing connection.

6.8 Diagnostic

SCPI Command :

```
DIAGnostic:SDBM
```

class DiagnosticCls

Diagnostic commands group definition. 79 total commands, 21 Subgroups, 1 group commands

set_sdbm(*text: str*) → None

```
# SCPI: DIAGnostic:SDBM
driver.diagnostic.set_sdbm(text = 'abc')
```

No command help available

param text
No help available

Cloning the Group

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

Subgroups

6.8.1 Base

class BaseCls

Base 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.diagnostic.base.clone()
```

Subgroups

6.8.1.1 Mmi

SCPI Command :

```
DIAGnostic:BASE:MMI:VERSion
```

class MmiCls

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

get_version() → List[str]

```
# SCPI: DIAGnostic:BASE:MMI:VERSion
value: List[str] = driver.diagnostic.base.mmi.get_version()
```

No command help available

return
version: No help available

6.8.1.2 Product

class ProductCls

Product 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.base.product.clone()
```

Subgroups

6.8.1.2.1 Option

class OptionCls

Option 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.base.product.option.clone()
```

Subgroups

6.8.1.2.1.1 Factory

SCPI Command :

```
DIAGnostic:BASE:PRODuct:OPTion:FACTory:CLEar
```

class FactoryCls

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

clear(*part_number: str*, *serial_number: str*) → None

```
# SCPI: DIAGnostic:BASE:PRODuct:OPTion:FACTory:CLEar
driver.diagnostic.base.product.option.factory.clear(part_number = 'abc', serial_
↪number = 'abc')
```

No command help available

param part_number
No help available

param serial_number
No help available

6.8.1.3 Salignment

class SalignmentCls

Salignment 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.diagnostic.base.salignment.clone()
```

Subgroups

6.8.1.3.1 Path

class PathCls

Path commands group definition. 6 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.base.salignment.path.clone()
```

Subgroups

6.8.1.3.1.1 Iq

SCPI Commands :

```
DIAGnostic:BASE:SALignment:PATH:IQ:STARt
DIAGnostic:BASE:SALignment:PATH:IQ:STATe
DIAGnostic:BASE:SALignment:PATH:IQ
```

class IqCls

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

get_start() → str

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:IQ:STARt
value: str = driver.diagnostic.base.salignment.path.iq.get_start()
```

No command help available

return
path: No help available

get_state() → List[str]

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:IQ:STATe
value: List[str] = driver.diagnostic.base.salignment.path.iq.get_state()
```

No command help available

return
state: No help available

get_value() → List[str]

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:IQ
value: List[str] = driver.diagnostic.base.salignment.path.iq.get_value()
```

No command help available

return
path_list: No help available

set_start(*path: str*) → None

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:IQ:STARt
driver.diagnostic.base.salignment.path.iq.set_start(path = 'abc')
```

No command help available

param path
No help available

6.8.1.3.1.2 Level

SCPI Commands :

```
DIAGnostic:BASE:SALignment:PATH:LEVel:STARt
DIAGnostic:BASE:SALignment:PATH:LEVel:STATe
DIAGnostic:BASE:SALignment:PATH:LEVel
```

class LevelCls

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

get_start() → str

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:LEVel:STARt
value: str = driver.diagnostic.base.salignment.path.level.get_start()
```

No command help available

return
path: No help available

get_state() → List[str]

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:LEVel:STATe
value: List[str] = driver.diagnostic.base.salignment.path.level.get_state()
```

No command help available

return
state: No help available

get_value() → List[str]

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:LEVel
value: List[str] = driver.diagnostic.base.salignment.path.level.get_value()
```

No command help available

return
path_list: No help available

set_start(*path: str*) → None

```
# SCPI: DIAGnostic:BASE:SALignment:PATH:LEVel:STARt
driver.diagnostic.base.salignment.path.level.set_start(path = 'abc')
```

No command help available

param path
No help available

6.8.2 BgInfo

SCPI Command :

```
DIAGnostic:BGINfo:CATalog
```

class BgInfoCls

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

get_catalog() → List[str]

```
# SCPI: DIAGnostic:BGINfo:CATalog
value: List[str] = driver.diagnostic.bgInfo.get_catalog()
```

No command help available

return
boards: No help available

6.8.3 Catalog

class CatalogCls

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

Cloning the Group

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

Subgroups

6.8.3.1 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.diagnostic.catalog.system.clone()
```

Subgroups

6.8.3.1.1 Connectors

SCPI Command :

```
DIAGnostic:CATalog:SYSTem:CONNectors
```

class ConnectorsCls

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

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

```
# SCPI: DIAGnostic:CATalog:SYSTem:CONNectors
value: List[str] = driver.diagnostic.catalog.system.connectors.get(name_style =␣
↪enums.NameStyle.FQName)
```

No command help available

param name_style
No help available

return
name_connector: No help available

6.8.4 Cmw<CmwVariant>

RepCap Settings

```
# Range: Cmw1 .. Cmw100
rc = driver.diagnostic.cmw.repcap_cmwVariant_get()
driver.diagnostic.cmw.repcap_cmwVariant_set(repcap.CmwVariant.Cmw1)
```

class CmwCls

Cmw commands group definition. 2 total commands, 1 Subgroups, 0 group commands Repeated Capability: CmwVariant, default value after init: CmwVariant.Cmw1

Cloning the Group

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

Subgroups

6.8.4.1 LedTest

class LedTestCls

LedTest 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.diagnostic.cmw.ledTest.clone()
```

Subgroups

6.8.4.1.1 Rx

SCPI Command :

```
DIAGnostic:CMW<variant>:LEDTest:RX
```

class RxCls

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

set(*test: bool*, *cmwVariant=CmwVariant.Default*) → None

```
# SCPI: DIAGnostic:CMW<variant>:LEDTest:RX
driver.diagnostic.cmw.ledTest.rx.set(test = False, cmwVariant = repcap.
↪CmwVariant.Default)
```

No command help available

param test
No help available

param cmwVariant
optional repeated capability selector. Default value: Cmw1 (settable in the interface 'Cmw')

6.8.4.1.2 Tx

SCPI Command :

```
DIAGnostic:CMW<variant>:LEDTest:TX
```

class TxCls

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

set(*test: bool*, *cmwVariant=CmwVariant.Default*) → None

```
# SCPI: DIAGnostic:CMW<variant>:LEDTest:TX
driver.diagnostic.cmw.ledTest.tx.set(test = False, cmwVariant = repcap.
↪CmwVariant.Default)
```

No command help available

param test
No help available

param cmwVariant
optional repeated capability selector. Default value: Cmw1 (settable in the interface 'Cmw')

6.8.5 Compass

SCPI Commands :

```
DIAGnostic:COMPass:VERSion
DIAGnostic:COMPass:HEAPcheck
```

class CompassCls

Compass commands group definition. 12 total commands, 3 Subgroups, 2 group commands

get_heap_check() → bool

```
# SCPI: DIAGnostic:COMPass:HEAPcheck
value: bool = driver.diagnostic.compass.get_heap_check()
```

No command help available

return
enable: No help available

get_version() → str

```
# SCPI: DIAGnostic:COMPass:VERSion
value: str = driver.diagnostic.compass.get_version()
```

No command help available

return
version: No help available

`set_heap_check`(*enable: bool*) → None

```
# SCPI: DIAGnostic:COMPass:HEAPcheck
driver.diagnostic.compass.set_heap_check(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.diagnostic.compass.clone()
```

Subgroups

6.8.5.1 Dbase

class DbaseCls

Dbase 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.diagnostic.compass.dbase.clone()
```

Subgroups

6.8.5.1.1 Rlogging

SCPI Commands :

```
DIAGnostic:COMPass:DBASe:RLOGging:MODE
DIAGnostic:COMPass:DBASe:RLOGging:DEVice
DIAGnostic:COMPass:DBASe:RLOGging:CLEar
```

class RloggingCls

Rlogging commands group definition. 4 total commands, 1 Subgroups, 3 group commands

clear() → None

```
# SCPI: DIAGnostic:COMPass:DBASe:RLOGging:CLEar
driver.diagnostic.compass.dbase.rlogging.clear()
```

No command help available

clear_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: DIAGnostic:COMPass:DBASe:RLOGging:CLEar
driver.diagnostic.compass.dbase.rlogging.clear_with_opc()
```

No command help available

Same as clear, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

get_device() → DiagLoggingDevice

```
# SCPI: DIAGnostic:COMPass:DBASe:RLOGging:DEVice
value: enums.DiagLoggingDevice = driver.diagnostic.compass.dbase.rlogging.get_
↪device()
```

No command help available

return
arg_0: No help available

get_mode() → DiagLoggigMode

```
# SCPI: DIAGnostic:COMPass:DBASe:RLOGging:MODE
value: enums.DiagLoggigMode = driver.diagnostic.compass.dbase.rlogging.get_
↪mode()
```

No command help available

return
arg_0: No help available

set_device(*arg_0: DiagLoggingDevice*) → None

```
# SCPI: DIAGnostic:COMPass:DBASe:RLOGging:DEVice
driver.diagnostic.compass.dbase.rlogging.set_device(arg_0 = enums.
↪DiagLoggingDevice.ALL)
```

No command help available

param arg_0
No help available

set_mode(*arg_0: DiagLoggigMode*) → None

```
# SCPI: DIAGnostic:COMPass:DBASe:RLOGging:MODE
driver.diagnostic.compass.dbase.rlogging.set_mode(arg_0 = enums.DiagLoggigMode.
↪DETailed)
```

No command help available

param arg_0
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.compass.dbase.rlogging.clone()
```

Subgroups

6.8.5.1.1.1 Protocol

SCPI Command :

```
DIAGnostic:COMPass:DBASe:RLOGging:PROTocol
```

class ProtocolCls

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

get(*file: str*) → str

```
# SCPI: DIAGnostic:COMPass:DBASe:RLOGging:PROTocol
value: str = driver.diagnostic.compass.dbase.rlogging.protocol.get(file = 'abc')
```

No command help available

param file
No help available

return
protocol: No help available

6.8.5.1.2 TaLogging

SCPI Commands :

```
DIAGnostic:COMPass:DBASe:TALogging:CLEar
DIAGnostic:COMPass:DBASe:TALogging:DEVice
```

class TaLoggingCls

TaLogging commands group definition. 4 total commands, 2 Subgroups, 2 group commands

clear() → None

```
# SCPI: DIAGnostic:COMPass:DBASe:TALogging:CLEar
driver.diagnostic.compass.dbase.taLogging.clear()
```

No command help available

clear_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: DIAGnostic:COMPass:DBASe:TALogging:CLEar
driver.diagnostic.compass.dbase.taLogging.clear_with_opc()
```

No command help available

Same as clear, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

get_device() → DiagLoggingDevice

```
# SCPI: DIAGnostic:COMPass:DBASe:TALogging:DEVice
value: enums.DiagLoggingDevice = driver.diagnostic.compass.dbase.taLogging.get_
↪device()
```

No command help available

return
device: No help available

set_device(*device: DiagLoggingDevice*) → None

```
# SCPI: DIAGnostic:COMPass:DBASe:TALogging:DEVice
driver.diagnostic.compass.dbase.taLogging.set_device(device = enums.
↪DiagLoggingDevice.ALL)
```

No command help available

param device
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.diagnostic.compass.dbase.taLogging.clone()
```

Subgroups

6.8.5.1.2.1 Mode

SCPI Command :

```
DIAGnostic:COMPass:DBASe:TALogging:MODE
```

class ModeCls

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

get(*name: str*) → DiagLoggigMode

```
# SCPI: DIAGnostic:COMPass:DBASe:TALogging:MODE
value: enums.DiagLoggigMode = driver.diagnostic.compass.dbase.taLogging.mode.
↪get(name = 'abc')
```

No command help available

param name
No help available

return
mod: No help available

set(*name: str*, *mod: DiagLoggigMode*) → None

```
# SCPI: DIAGnostic:COMPass:DBASe:TALogging:MODE
driver.diagnostic.compass.dbase.taLogging.mode.set(name = 'abc', mod = enums.
↪DiagLoggigMode.DETailed)
```

No command help available

param name
No help available

param mod
No help available

6.8.5.1.2.2 Protocol

SCPI Command :

```
DIAGnostic:COMPass:DBASe:TALogging:PROTocol
```

class ProtocolCls

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

get(*file: str*) → int

```
# SCPI: DIAGnostic:COMPass:DBASe:TALogging:PROTocol
value: int = driver.diagnostic.compass.dbase.taLogging.protocol.get(file = 'abc
↪')
```

No command help available

param file
No help available

return
result: No help available

6.8.5.2 Debug

SCPI Command :

```
DIAGnostic:COMPass:DEBug:MODE
```

class DebugCls

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

get_mode() → bool

```
# SCPI: DIAGnostic:COMPass:DEBug:MODE
value: bool = driver.diagnostic.compass.debug.get_mode()
```

No command help available

return
debug_mode: No help available

set_mode(*debug_mode: bool*) → None

```
# SCPI: DIAGnostic:COMPass:DEBug:MODE
driver.diagnostic.compass.debug.set_mode(debug_mode = False)
```

No command help available

param debug_mode
No help available

6.8.5.3 Statistics

class StatisticsCls

Statistics 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.compass.statistics.clone()
```

Subgroups

6.8.5.3.1 Process

SCPI Command :

```
DIAGnostic:COMPass:STATistics:PROCess
```

class ProcessCls

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

get(*token: str*) → str

```
# SCPI: DIAGnostic:COMPass:STATistics:PROCess
value: str = driver.diagnostic.compass.statistics.process.get(token = 'abc')
```

No command help available

param token
No help available

return
statistics: No help available

6.8.6 Configure

class ConfigureCls

Configure commands group definition. 18 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.8.6.1 System

class SystemCls

System 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.diagnostic.configure.system.clone()
```

Subgroups

6.8.6.1.1 Dapi

class DapiCls

Dapi commands group definition. 16 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.8.6.1.1.1 Logging

class LoggingCls

Logging commands group definition. 16 total commands, 2 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.8.6.1.1.2 File

class FileCls

File 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.diagnostic.configure.system.dapi.logging.file.clone()
```

Subgroups

6.8.6.1.1.3 Psub

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:PAYLoad
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB
```

class PsubCls

Psub commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_payload() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:PAYLoad
value: bool = driver.diagnostic.configure.system.dapi.logging.file.psub.get_
↪payload()
```

No command help available

return
payload: No help available

get_value() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB
value: bool = driver.diagnostic.configure.system.dapi.logging.file.psub.get_
↪value()
```

No command help available

return
enable: No help available

set_payload(*payload: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:PAYLoad
driver.diagnostic.configure.system.dapi.logging.file.psub.set_payload(payload =␣
↪False)
```

No command help available

param payload
No help available

set_value(*enable: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB
driver.diagnostic.configure.system.dapi.logging.file.psub.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.diagnostic.configure.system.dapi.logging.file.psub.clone()
```

Subgroups

6.8.6.1.1.4 FilterPy

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:FILTer:MNAMe
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:FILTer:RNAMe
```

class FilterPyCls

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

get_mname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:FILTer:MNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.file.psub.filterPy.
↪get_mname()
```

No command help available

return
filter_mname: No help available

get_rname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:FILTer:RNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.file.psub.filterPy.
↪get_rname()
```

No command help available

return
filter_rname: No help available

set_mname(*filter_mname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:FILTer:MNAMe
driver.diagnostic.configure.system.dapi.logging.file.psub.filterPy.set_
↪mname(filter_mname = 'abc')
```

No command help available

param filter_mname
No help available

set_rname(*filter_rname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:PSUB:FILTer:RNAMe
driver.diagnostic.configure.system.dapi.logging.file.psub.filterPy.set_
↪rname(filter_rname = 'abc')
```

No command help available

param filter_rname
No help available

6.8.6.1.1.5 Rpc

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:PAYLoad
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC
```

class RpcCls

Rpc commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_payload() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:PAYLoad
value: bool = driver.diagnostic.configure.system.dapi.logging.file.rpc.get_
↪payload()
```

No command help available

return
payload: No help available

get_value() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC
value: bool = driver.diagnostic.configure.system.dapi.logging.file.rpc.get_
↪value()
```

No command help available

return
enable: No help available

set_payload(*payload: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:PAYLoad
driver.diagnostic.configure.system.dapi.logging.file.rpc.set_payload(payload =␣
↪False)
```

No command help available

param payload
No help available

set_value(*enable: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC
driver.diagnostic.configure.system.dapi.logging.file.rpc.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.diagnostic.configure.system.dapi.logging.file.rpc.clone()
```

Subgroups

6.8.6.1.1.6 FilterPy

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:FILTer:MNAMe
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:FILTer:RNAMe
```

class FilterPyCls

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

get_mname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:FILTer:MNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.file.rpc.filterPy.
↪get_mname()
```

No command help available

return
filter_mname: No help available

get_rname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:FILTer:RNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.file.rpc.filterPy.
↪get_rname()
```

No command help available

return
filter_rname: No help available

set_mname(*filter_mname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:FILTer:MNAMe
driver.diagnostic.configure.system.dapi.logging.file.rpc.filterPy.set_
↪mname(filter_mname = 'abc')
```

No command help available

param filter_mname
No help available

set_rname(*filter_rname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:FILE:RPC:FILTer:RNAMe
driver.diagnostic.configure.system.dapi.logging.file.rpc.filterPy.set_
↪rname(filter_rname = 'abc')
```

No command help available

param filter_rname
No help available

6.8.6.1.1.7 Mars

class MarsCls

Mars 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.diagnostic.configure.system.dapi.logging.mars.clone()
```

Subgroups

6.8.6.1.1.8 Psub

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:PAYLoad
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB
```

class PsubCls

Psub commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_payload() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:PAYLoad
value: bool = driver.diagnostic.configure.system.dapi.logging.mars.psub.get_
↪payload()
```

No command help available

return
payload: No help available

get_value() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB
value: bool = driver.diagnostic.configure.system.dapi.logging.mars.psub.get_
↪value()
```

No command help available

return
enable: No help available

set_payload(*payload: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:PAYLoad
driver.diagnostic.configure.system.dapi.logging.mars.psub.set_payload(payload =␣
↪False)
```

No command help available

param payload
No help available

set_value(*enable: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB
driver.diagnostic.configure.system.dapi.logging.mars.psub.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.diagnostic.configure.system.dapi.logging.mars.psub.clone()
```

Subgroups

6.8.6.1.1.9 FilterPy

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:FILTer:MNAMe
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:FILTer:RNAMe
```

class FilterPyCls

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

get_mname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:FILTer:MNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.mars.psub.filterPy.
↪get_mname()
```

No command help available

return
filter_mname: No help available

get_rname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:FILTer:RNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.mars.psub.filterPy.
↪get_rname()
```

No command help available

return
filter_rname: No help available

set_mname(*filter_mname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:FILTer:MNAMe
driver.diagnostic.configure.system.dapi.logging.mars.psub.filterPy.set_
↪mname(filter_mname = 'abc')
```

No command help available

param filter_mname
No help available

set_rname(*filter_rname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:PSUB:FILTer:RNAMe
driver.diagnostic.configure.system.dapi.logging.mars.psub.filterPy.set_
↪rname(filter_rname = 'abc')
```

No command help available

param filter_rname
No help available

6.8.6.1.1.10 Rpc

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:PAYLoad
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC
```

class RpcCls

Rpc commands group definition. 4 total commands, 1 Subgroups, 2 group commands

get_payload() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:PAYLoad
value: bool = driver.diagnostic.configure.system.dapi.logging.mars.rpc.get_
↪payload()
```

No command help available

return
payload: No help available

get_value() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC
value: bool = driver.diagnostic.configure.system.dapi.logging.mars.rpc.get_
↪value()
```

No command help available

return
enable: No help available

set_payload(*payload: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:PAYLoad
driver.diagnostic.configure.system.dapi.logging.mars.rpc.set_payload(payload =␣
↪False)
```

No command help available

param payload
No help available

set_value(*enable: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC
driver.diagnostic.configure.system.dapi.logging.mars.rpc.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.diagnostic.configure.system.dapi.logging.mars.rpc.clone()
```

Subgroups

6.8.6.1.1.11 FilterPy

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:FILTer:MNAMe
DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:FILTer:RNAMe
```

class FilterPyCls

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

get_mname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:FILTer:MNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.mars.rpc.filterPy.
↪get_mname()
```

No command help available

return
filter_mname: No help available

get_rname() → str

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:FILTer:RNAMe
value: str = driver.diagnostic.configure.system.dapi.logging.mars.rpc.filterPy.
↪get_rname()
```

No command help available

return
filter_rname: No help available

set_mname(*filter_mname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:FILTer:MNAMe
driver.diagnostic.configure.system.dapi.logging.mars.rpc.filterPy.set_
↪mname(filter_mname = 'abc')
```

No command help available

param filter_mname
No help available

set_rname(*filter_rname: str*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:DAPI:LOGGing:MARS:RPC:FILTer:RNAMe
driver.diagnostic.configure.system.dapi.logging.mars.rpc.filterPy.set_
↪rname(filter_rname = 'abc')
```

No command help available

param filter_rname
No help available

6.8.6.1.2 Scpi

class ScpiCls

Scpi 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.configure.system.scpi.clone()
```

Subgroups

6.8.6.1.2.1 Logging

SCPI Commands :

```
DIAGnostic[:CONFigure]:SYSTem:SCPI:LOGGing:FILE
DIAGnostic[:CONFigure]:SYSTem:SCPI:LOGGing:MARS
```

class LoggingCls

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

get_file() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:SCPI:LOGGing:FILE
value: bool = driver.diagnostic.configure.system.scpi.logging.get_file()
```

No command help available

return
enable: No help available

get_mars() → bool

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:SCPI:LOGGing:MARS
value: bool = driver.diagnostic.configure.system.scpi.logging.get_mars()
```

No command help available

return
enable: No help available

set_file(*enable: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:SCPI:LOGGing:FILE
driver.diagnostic.configure.system.scpi.logging.set_file(enable = False)
```

No command help available

param enable
No help available

set_mars(*enable: bool*) → None

```
# SCPI: DIAGnostic[:CONFigure]:SYSTem:SCPI:LOGGing:MARS
driver.diagnostic.configure.system.scpi.logging.set_mars(enable = False)
```

No command help available

param enable
No help available

6.8.7 Eeprom

class EepromCls

Eeprom 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.diagnostic.eeprom.clone()
```

Subgroups

6.8.7.1 Data

SCPI Command :

```
DIAGnostic:EEPRom:DATA
```

class DataCls

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

get() → str

```
# SCPI: DIAGnostic:EEPRom:DATA
value: str = driver.diagnostic.eeprom.data.get()
```

No command help available

return
data_folder: No help available

set(*board: str*, *table_id: int*, *board_instance: int = None*) → None

```
# SCPI: DIAGnostic:EEPRom:DATA
driver.diagnostic.eeprom.data.set(board = 'abc', table_id = 1, board_instance =␣
↪1)
```

No command help available

param board
No help available

param table_id
No help available

param board_instance
No help available

6.8.7.2 Header

SCPI Command :

```
DIAGnostic:EEPRom:HEADer
```

class HeaderCls

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

get() → str

```
# SCPI: DIAGnostic:EEPRom:HEADer
value: str = driver.diagnostic.eeprom.header.get()
```

No command help available

return
header: No help available

set(*board: str*, *board_instance: int = None*) → None

```
# SCPI: DIAGnostic:EEPRom:HEADer
driver.diagnostic.eeprom.header.set(board = 'abc', board_instance = 1)
```

No command help available

param board
No help available

param board_instance
No help available

6.8.8 Error

class ErrorCls

Error 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.error.clone()
```

Subgroups

6.8.8.1 Queue

SCPI Commands :

```
DIAGnostic:ERRor:QUEue:SIZE
DIAGnostic:ERRor:QUEue:LENGth
```

class QueueCls

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

get_length() → int

```
# SCPI: DIAGnostic:ERRor:QUEue:LENGth
value: int = driver.diagnostic.error.queue.get_length()
```

No command help available

return
text_length: No help available

get_size() → int

```
# SCPI: DIAGnostic:ERRor:QUEue:SIZE
value: int = driver.diagnostic.error.queue.get_size()
```

No command help available

return
size: No help available

set_length(*text_length: int*) → None

```
# SCPI: DIAGnostic:ERRor:QUEue:LENGth
driver.diagnostic.error.queue.set_length(text_length = 1)
```

No command help available

param text_length
No help available

set_size(*size: int*) → None

```
# SCPI: DIAGnostic:ERRor:QUEue:SIZE
driver.diagnostic.error.queue.set_size(size = 1)
```

No command help available

param size
No help available

Cloning the Group

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

Subgroups

6.8.8.1.1 Push

SCPI Command :

```
DIAGnostic:ERRor:QUEue:PUSH
```

class PushCls

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

set(*code: int*, *text: str*, *guid: str*, *info: str*) → None

```
# SCPI: DIAGnostic:ERRor:QUEue:PUSH
driver.diagnostic.error.queue.push.set(code = 1, text = 'abc', guid = 'abc',␣
↪info = 'abc')
```

No command help available

param code
No help available

param text
No help available

param guid
No help available

param info
No help available

6.8.9 Fetch

class FetchCls

Fetch 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.fetch.clone()
```

Subgroups

6.8.9.1 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.diagnostic.fetch.power.clone()
```

Subgroups

6.8.9.1.1 State

SCPI Command :

```
DIAGnostic:FETCh:POWer:STATe
```

class StateCls

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

class GetStruct

Response structure. Fields:

- Main_State: enums.TargetStateA: No parameter help available
- Synch_State: enums.TargetSyncState: No parameter help available

get(*timeout: float = None*, *target_main_state: TargetStateA = None*, *target_sync_state: TargetSyncState = None*) → GetStruct

```
# SCPI: DIAGnostic:FETCh:POWer:STATe
value: GetStruct = driver.diagnostic.fetch.power.state.get(timeout = 1.0,␣
↪target_main_state = enums.TargetStateA.OFF, target_sync_state = enums.
↪TargetSyncState.ADJusted)
```

No command help available

param timeout
No help available

param target_main_state
No help available

param target_sync_state
No help available

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

6.8.10 FootPrint

class FootPrintCls

FootPrint 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.diagnostic.footPrint.clone()
```

Subgroups

6.8.10.1 Element

SCPI Command :

```
DIAGnostic:FOOTprint:ELEMent:IDS
```

class ElementCls

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

get_ids() → List[int]

```
# SCPI: DIAGnostic:FOOTprint:ELEMent:IDS
value: List[int] = driver.diagnostic.footPrint.element.get_ids()
```

No command help available

return
element_ids: No help available

Cloning the Group

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

Subgroups

6.8.10.1.1 Connection

class ConnectionCls

Connection 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.footPrint.element.connection.clone()
```

Subgroups

6.8.10.1.1.1 Target

class TargetCls

Target 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.footPrint.element.connection.target.clone()
```

Subgroups

6.8.10.1.1.2 Ids

SCPI Command :

```
DIAGnostic:FOOTprint:ELEMent:CONNection:TARGet:IDS
```

class IdsCls

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

get(*ielement_id: float*) → str

```
# SCPI: DIAGnostic:FOOTprint:ELEMent:CONNection:TARGet:IDS
value: str = driver.diagnostic.footPrint.element.connection.target.ids.
↪get(ielement_id = 1.0)
```

No command help available

param ielement_id
No help available

return
starget_ids: No help available

6.8.10.1.2 Data

SCPI Command :

```
DIAGnostic:FOOTprint:ELEMent:DATA
```

class DataCls

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

class GetStruct

Response structure. Fields:

- Stype_Name: str: No parameter help available
- Spart_Name: str: No parameter help available
- Efunctionality: int: No parameter help available

get(*ielement_id: float*) → GetStruct

```
# SCPI: DIAGnostic:FOOTprint:ELEMent:DATA
value: GetStruct = driver.diagnostic.footPrint.element.data.get(ielement_id = 1.
↪0)
```

No command help available

param ielement_id
No help available

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

6.8.10.1.3 Properties

SCPI Command :

```
DIAGnostic:FOOTprint:ELEMent:PROPerties
```

class PropertiesCls

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

get(*ielement_id: float*, *bnormalized: bool*) → str

```
# SCPI: DIAGnostic:FOOTprint:ELEMent:PROPerties
value: str = driver.diagnostic.footPrint.element.properties.get(ielement_id = 1.
↪0, bnormalized = False)
```

No command help available

param ielement_id
No help available

param bnormalized
No help available

return
sproperties: No help available

6.8.10.1.4 References

SCPI Command :

```
DIAGnostic:FOOTprint:ELEMent:REFerences
```

class ReferencesCls

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

get(*ielement_id: float*, *sreference_type: str*) → List[int]

```
# SCPI: DIAGnostic:FOOTprint:ELEMent:REFerences
value: List[int] = driver.diagnostic.footPrint.element.references.get(ielement_
↪id = 1.0, sreference_type = 'abc')
```

No command help available

param ielement_id
No help available

param sreference_type
No help available

return
ielement_ids: No help available

6.8.10.2 Li

class LiCls

Li 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.footPrint.li.clone()
```

Subgroups

6.8.10.2.1 Usecases

SCPI Command :

```
DIAGnostic:FOOTprint:LI:USECases
```

class UsecasesCls

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

get(*ili_id: int*) → str

```
# SCPI: DIAGnostic:FOOTprint:LI:USECases
value: str = driver.diagnostic.footPrint.li.usecases.get(ili_id = 1)
```

No command help available

param ili_id
No help available

return
the_result: No help available

6.8.10.3 UseCase

SCPI Command :

```
DIAGnostic:FOOTprint:USECase:IDS
```

class UseCaseCls

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

get_ids() → List[int]

```
# SCPI: DIAGnostic:FOOTprint:USECase:IDS
value: List[int] = driver.diagnostic.footPrint.useCase.get_ids()
```

No command help available

return
ids: No help available

Cloning the Group

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

Subgroups

6.8.10.3.1 Data

SCPI Command :

```
DIAGnostic:FOOTprint:USECase:DATA
```

class DataCls

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

class GetStruct

Response structure. Fields:

- Sname: str: No parameter help available
- Efunctionality: int: No parameter help available

get(*iuse_case_id: int*) → GetStruct

```
# SCPI: DIAGnostic:FOOTprint:USECase:DATA
value: GetStruct = driver.diagnostic.footPrint.useCase.data.get(iuse_case_id =␣
↪1)
```

No command help available

param iuse_case_id
No help available

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

6.8.11 Generic

class GenericCls

Generic 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.generic.clone()
```

Subgroups

6.8.11.1 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.diagnostic.generic.measurement.clone()
```

Subgroups

6.8.11.1.1 Dapi

SCPI Command :

```
DIAGnostic:GENeric:MEASurement:DAPI:TOUT
```

class DapiCls

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

get_timeout() → float

```
# SCPI: DIAGnostic:GENeric:MEASurement:DAPI:TOUT
value: float = driver.diagnostic.generic.measurement.dapi.get_timeout()
```

No command help available

return
timeout: No help available

set_timeout(*timeout: float*) → None

```
# SCPI: DIAGnostic:GENeric:MEASurement:DAPI:TOUT
driver.diagnostic.generic.measurement.dapi.set_timeout(timeout = 1.0)
```

No command help available

param timeout
No help available

6.8.12 Help

SCPI Command :

```
DIAGnostic:HELP:HEADers
```

class HelpCls

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

get_headers() → bytes

```
# SCPI: DIAGnostic:HELP:HEADers
value: bytes = driver.diagnostic.help.get_headers()
```

No command help available

return
header: No help available

Cloning the Group

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

Subgroups

6.8.12.1 Syntax

SCPI Commands :

```
DIAGnostic:HELP:SYNTax
DIAGnostic:HELP:SYNTax:ALL
```

class SyntaxCls

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

get(*header: str*) → str

```
# SCPI: DIAGnostic:HELP:SYNTax
value: str = driver.diagnostic.help.syntax.get(header = 'abc')
```

No command help available

param header
No help available

return
syntax: No help available

get_all() → bytes

```
# SCPI: DIAGnostic:HELP:SYNTax:ALL
value: bytes = driver.diagnostic.help.syntax.get_all()
```

No command help available

return
commands: No help available

6.8.13 Instrument

SCPI Commands :

```
DIAGnostic:INSTrument:LOAD
DIAGnostic:INSTrument:UNLoad
```

class InstrumentCls

Instrument commands group definition. 4 total commands, 2 Subgroups, 2 group commands

load(*appl_name_and_li_number: str*) → None

```
# SCPI: DIAGnostic:INSTrument:LOAD
driver.diagnostic.instrument.load(appl_name_and_li_number = 'abc')
```

No command help available

param appl_name_and_li_number
No help available

set_unload(*appl_name_and_li_number: str*) → None

```
# SCPI: DIAGnostic:INSTrument:UNLoad
driver.diagnostic.instrument.set_unload(appl_name_and_li_number = 'abc')
```

No command help available

param appl_name_and_li_number
No help available

Cloning the Group

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

Subgroups

6.8.13.1 Application

class ApplicationCls

Application 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.instrument.application.clone()
```

Subgroups

6.8.13.1.1 Count

SCPI Command :

```
DIAGnostic:INSTrument:APPLication:COUNt
```

class CountCls

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

get(*fw_app_name: str*) → int

```
# SCPI: DIAGnostic:INSTrument:APPLication:COUNt
value: int = driver.diagnostic.instrument.application.count.get(fw_app_name =
↪'abc')
```

No command help available

param fw_app_name
No help available

return
instance_count: No help available

6.8.13.2 Consistency

SCPI Command :

```
DIAGnostic:INSTrument:CONSistency
```

class ConsistencyCls

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

get(*appl_name: str*) → int

```
# SCPI: DIAGnostic:INSTrument:CONSistency
value: int = driver.diagnostic.instrument.consistency.get(appl_name = 'abc')
```

No command help available

param appl_name
No help available

return
consistent: No help available

6.8.14 Log

class LogCls

Log 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.log.clone()
```

Subgroups

6.8.14.1 Dump

SCPI Command :

```
DIAGnostic:LOG:DUMP
```

class DumpCls

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

set() → None

```
# SCPI: DIAGnostic:LOG:DUMP
driver.diagnostic.log.dump.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: DIAGnostic:LOG:DUMP
driver.diagnostic.log.dump.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.8.15 Meas

class MeasCls

Meas 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.meas.clone()
```

Subgroups

6.8.15.1 Scpi

SCPI Commands :

```
DIAGnostic:MEAS:SCPI:VERSion
DIAGnostic:MEAS:SCPI:HOST
```

class ScpiCls

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

get_host() → str

```
# SCPI: DIAGnostic:MEAS:SCPI:HOST
value: str = driver.diagnostic.meas.scpi.get_host()
```

No command help available

return
name: No help available

get_version() → float

```
# SCPI: DIAGnostic:MEAS:SCPI:VERSion
value: float = driver.diagnostic.meas.scpi.get_version()
```

No command help available

return
version: No help available

6.8.16 Record

class RecordCls

Record 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.record.clone()
```

Subgroups

6.8.16.1 Macro

class MacroCls

Macro 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.record.macro.clone()
```

Subgroups

6.8.16.1.1 File

SCPI Commands :

```
DIAGnostic:RECord:MACRo:FILE:SIZE
DIAGnostic:RECord:MACRo:FILE:FILTer
```

class FileCls

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

class FilterPyStruct

Structure for setting input parameters. Fields:

- Binput: bool: No parameter help available
- Boutput: bool: No parameter help available
- Berror: bool: No parameter help available
- Btrigger: bool: No parameter help available
- Bdevice_Clear: bool: No parameter help available
- Bstatus_Register: bool: No parameter help available
- Bconnection: bool: No parameter help available
- Bremote_Local_Events: bool: No parameter help available

get_filter_py() → FilterPyStruct

```
# SCPI: DIAGnostic:RECord:MACRo:FILE:FILTer
value: FilterPyStruct = driver.diagnostic.record.macro.file.get_filter_py()
```

No command help available

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

get_size() → int

```
# SCPI: DIAGnostic:RECord:MACRo:FILE:SIZE
value: int = driver.diagnostic.record.macro.file.get_size()
```

No command help available

return
ifile_size: No help available

set_filter_py(*value: FilterPyStruct*) → None

```
# SCPI: DIAGnostic:RECord:MACRo:FILE:FILTer
structure = driver.diagnostic.record.macro.file.FilterPyStruct()
structure.Binput: bool = False
structure.Boutput: bool = False
structure.Berror: bool = False
structure.Btrigger: bool = False
structure.Bdevice_Clear: bool = False
structure.Bstatus_Register: bool = False
structure.Bconnection: bool = False
structure.Bremote_Local_Events: bool = False
driver.diagnostic.record.macro.file.set_filter_py(value = structure)
```

No command help available

param value
see the help for FilterPyStruct structure arguments.

set_size(*ifile_size: int*) → None

```
# SCPI: DIAGnostic:RECord:MACRo:FILE:SIZE
driver.diagnostic.record.macro.file.set_size(ifile_size = 1)
```

No command help available

param ifile_size
No help available

6.8.17 Route

class RouteCls

Route 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.diagnostic.route.clone()
```

Subgroups

6.8.17.1 Gprf

class GprfCls

Gprf 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.diagnostic.route.gprf.clone()
```

Subgroups

6.8.17.1.1 Generator

SCPI Command :

```
DIAGnostic:ROUTe:GPRF:GENerator<Instance>:SPATh
```

class GeneratorCls

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

get_spath() → List[str]

```
# SCPI: DIAGnostic:ROUTe:GPRF:GENerator<Instance>:SPATh
value: List[str] = driver.diagnostic.route.gprf.generator.get_spath()
```

No command help available

return
signal_path: No help available

set_spath(*signal_path: List[str]*) → None

```
# SCPI: DIAGnostic:ROUTe:GPRF:GENerator<Instance>:SPATh
driver.diagnostic.route.gprf.generator.set_spath(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.8.17.1.2 Measurement

SCPI Command :

```
DIAGnostic:ROUTe:GPRF:MEASurement<Instance>:SPATh
```

class MeasurementCls

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

get_spath() → List[str]

```
# SCPI: DIAGnostic:ROUTe:GPRF:MEASurement<Instance>:SPATh
value: List[str] = driver.diagnostic.route.gprf.measurement.get_spath()
```

No command help available

return
signal_path: No help available

set_spath(*signal_path: List[str]*) → None

```
# SCPI: DIAGnostic:ROUTe:GPRF:MEASurement<Instance>:SPATh
driver.diagnostic.route.gprf.measurement.set_spath(signal_path = ['abc1', 'abc2
↪', 'abc3'])
```

No command help available

param signal_path
No help available

6.8.17.2 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.diagnostic.route.nrMmw.clone()
```

Subgroups

6.8.17.2.1 Measurement

SCPI Command :

```
DIAGnostic:ROUTe:NRMMw:MEASurement<Instance>:SPATh
```

class MeasurementCls

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

get_spath() → List[str]

```
# SCPI: DIAGnostic:ROUTe:NRMMw:MEASurement<Instance>:SPATh
value: List[str] = driver.diagnostic.route.nrMmw.measurement.get_spath()
```

No command help available

return
signal_path: No help available

set_spath(*signal_path: List[str]*) → None

```
# SCPI: DIAGnostic:ROUTe:NRMMw:MEASurement<Instance>:SPATh
driver.diagnostic.route.nrMmw.measurement.set_spath(signal_path = ['abc1', 'abc2
↪', 'abc3'])
```

No command help available

param signal_path
No help available

6.8.17.3 Uwb

class UwbCls

Uwb 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.route.uwb.clone()
```

Subgroups

6.8.17.3.1 Measurement

SCPI Command :

```
DIAGnostic:ROUTe:UWB:MEASurement<Instance>:SPATh
```

class MeasurementCls

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

get_spath() → List[str]

```
# SCPI: DIAGnostic:ROUTe:UWB:MEASurement<Instance>:SPATh
value: List[str] = driver.diagnostic.route.uwb.measurement.get_spath()
```

No command help available

return
signal_path: No help available

set_spath(*signal_path: List[str]*) → None

```
# SCPI: DIAGnostic:ROUTe:UWB:MEASurement<Instance>:SPATh
driver.diagnostic.route.uwb.measurement.set_spath(signal_path = ['abc1', 'abc2',
↪ 'abc3'])
```

No command help available

param signal_path
No help available

6.8.18 Routing

SCPI Command :

```
DIAGnostic:ROUTing:CATalog
```

class RoutingCls

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

get_catalog() → List[str]

```
# SCPI: DIAGnostic:ROUTing:CATalog
value: List[str] = driver.diagnostic.routing.get_catalog()
```

No command help available

return
routing_name: No help available

Cloning the Group

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

Subgroups

6.8.18.1 Expert

class ExpertCls

Expert 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.routing.expert.clone()
```

Subgroups

6.8.18.1.1 Setup

SCPI Command :

```
DIAGnostic:ROUTing:EXPert:SETup
```

class SetupCls

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

get(*routing_name: str*) → List[ExpertSetup]

```
# SCPI: DIAGnostic:ROUTing:EXPert:SETup
value: List[enums.ExpertSetup] = driver.diagnostic.routing.expert.setup.
↪get(routing_name = 'abc')
```

No command help available

param routing_name
No help available

return
data: No help available

set(*routing_name: str*, *data: List[ExpertSetup]*) → None

```
# SCPI: DIAGnostic:ROUTing:EXPert:SETup
driver.diagnostic.routing.expert.setup.set(routing_name = 'abc', data =␣
↪[ExpertSetup.BBG1, ExpertSetup.SUW7])
```

No command help available

param routing_name
No help available

param data
No help available

6.8.19 SingleCmw

SCPI Command :

```
DIAGnostic:CMWS:LEDTest
```

class SingleCmwCls

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

set_led_test(*test: bool*) → None

```
# SCPI: DIAGnostic:CMWS:LEDTest
driver.diagnostic.singleCmw.set_led_test(test = False)
```

No command help available

param test
No help available

6.8.20 Status

SCPI Command :

```
DIAGnostic:STATus:OPC
```

class StatusCls

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

class OpcStruct

Structure for reading output parameters. Fields:

- Opc_Counter: int: No parameter help available
- Opc_Active_Counter: int: No parameter help available
- Opc_State: bool: No parameter help available
- Opc_Command_State: bool: No parameter help available
- Opc_Query_State: bool: No parameter help available

get_opc() → OpcStruct

```
# SCPI: DIAGnostic:STATus:OPC
value: OpcStruct = driver.diagnostic.status.get_opc()
```

No command help available

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

6.8.21 Trigger

class TriggerCls

Trigger 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.trigger.clone()
```

Subgroups

6.8.21.1 Add

class AddCls

Add 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.trigger.add.clone()
```

Subgroups

6.8.21.1.1 Debug

class DebugCls

Debug 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.trigger.add.debug.clone()
```

Subgroups

6.8.21.1.1.1 Output

SCPI Command :

```
DIAGnostic:TRIGger:ADD:DEBug:OUTPut
```

class OutputCls

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

class OutputStruct

Response structure. Fields:

- Show_Trigger_Debug_Output: bool: No parameter help available
- Show_Trigger_Debug_Scpi_Output: bool: No parameter help available

get() → OutputStruct

```
# SCPI: DIAGnostic:TRIGger:ADD:DEBug:OUTPut
value: OutputStruct = driver.diagnostic.trigger.add.debug.output.get()
```

No command help available

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

set(*show_trigger_debug_output: bool*, *show_trigger_debug_scpi_output: bool*) → None

```
# SCPI: DIAGnostic:TRIGger:ADD:DEBug:OUTPut
driver.diagnostic.trigger.add.debug.output.set(show_trigger_debug_output =␣
↪False, show_trigger_debug_scpi_output = False)
```

No command help available

param show_trigger_debug_output
No help available

param show_trigger_debug_scpi_output
No help available

6.9 Display

SCPI Command :

```
DISPlay:FORMat
```

class DisplayCls

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

get_format_py() → str

```
# SCPI: DISPlay:FORMat
value: str = driver.display.get_format_py()
```

No command help available

return
arg_0: No help available

set_format_py(*arg_0: str*) → None

```
# SCPI: DISPlay:FORMat
driver.display.set_format_py(arg_0 = rawAbc)
```

No command help available

param arg_0
No help available

Cloning the Group

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

Subgroups

6.9.1 Window<Window>

RepCap Settings

```
# Range: Win1 .. Win32
rc = driver.display.window.repcap_window_get()
driver.display.window.repcap_window_set(repcap.Window.Win1)
```

class WindowCls

Window commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Window, default value after init: Window.Win1

Cloning the Group

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

Subgroups

6.9.1.1 Select

SCPI Command :

```
DISPlay[:WINDow<1-n>]:SELect
```

class SelectCls

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

set(*window=Window.Default*) → None

```
# SCPI: DISPlay[:WINDow<1-n>]:SELect
driver.display.window.select.set(window = repcap.Window.Default)
```

No command help available

param window
optional repeated capability selector. Default value: Win1 (settable in the interface 'Window')

set_with_opc(*window=Window.Default*, *opc_timeout_ms: int = -1*) → None

6.10 FirmwareUpdate

SCPI Command :

```
FETCh:FWUPdate:VERSions
```

class FirmwareUpdateCls

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

get_versions() → List[str]

```
# SCPI: FETCh:FWUPdate:VERSions
value: List[str] = driver.firmwareUpdate.get_versions()
```

No command help available

return
versions: No help available

6.11 FormatPy

class FormatPyCls

FormatPy commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.11.1 Base

SCPI Commands :

```
FORMat:BASE:BORDer
FORMat:BASE:DINTerchange
FORMat:BASE:SREGister
```

class BaseCls

Base commands group definition. 4 total commands, 1 Subgroups, 3 group commands

get_border() → ByteOrder

```
# SCPI: FORMat:BASE:BORDer
value: enums.ByteOrder = driver.formatPy.base.get_border()
```

No command help available

return
byte_order: No help available

get_dinterchange() → bool

```
# SCPI: FORMat:BASE:DINTerchange
value: bool = driver.formatPy.base.get_dinterchange()
```

No command help available

return
dif_format: No help available

get_sregister() → StatRegFormat

```
# SCPI: FORMat:BASE:SREGister
value: enums.StatRegFormat = driver.formatPy.base.get_sregister()
```

No command help available

return
status_register_format: No help available

set_border(*byte_order: ByteOrder*) → None

```
# SCPI: FORMat:BASE:BORDer
driver.formatPy.base.set_border(byte_order = enums.ByteOrder.NORMal)
```

No command help available

param byte_order
No help available

set_dinterchange(*dif_format: bool*) → None

```
# SCPI: FORMat:BASE:DINTerchange
driver.formatPy.base.set_dinterchange(dif_format = False)
```

No command help available

param dif_format
No help available

set_sregister(*status_register_format: StatRegFormat*) → None

```
# SCPI: FORMat:BASE:SREGister
driver.formatPy.base.set_sregister(status_register_format = enums.StatRegFormat.
↪ASCii)
```

No command help available

param status_register_format
No help available

Cloning the Group

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

Subgroups

6.11.1.1 Data

SCPI Command :

```
FORMat:BASE[:DATA]
```

class DataCls

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

class DataStruct

Response structure. Fields:

- **Data_Type: enums.DataFormat:**
  - ASCii: Numeric data is transferred as ASCII bytes. Floating point numbers are transferred in scientific E notation.
  - REAL: Numeric data is transferred in a definite length block as IEEE floating point numbers (block data) .
  - BINary | HEXadecimal | OCTal: Numeric data is transferred in binary, hexadecimal or octal format.
- Data_Length: int: The meaning depends on the DataType as listed below. A zero returned by a query means that the

default value is used.

- For ASCii: Decimal places of floating point numbers. That means, number of 'b' digits in the scientific notation a.bbbbbbE+ccc.Default: six decimal places
- For REAL: Length of floating point numbers in bits:32 bits = 4 bytes, format #14...64 bits = 8 bytes, format #18...Default: 64 bits
- For BINary, HEXadecimal, OCTal: Minimum number of digits. If the number is longer, more digits are used. If it is shorter, leading zeros are added.Default: 0, no leading zeros

get() → DataStruct

```
# SCPI: FORMat:BASE[:DATA]
value: DataStruct = driver.formatPy.base.data.get()
```

Selects the format for numeric data transferred to and from the CMX500, for example query results.

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

set(*data_type: DataFormat*, *data_length: int = None*) → None

```
# SCPI: FORMat:BASE[:DATA]
driver.formatPy.base.data.set(data_type = enums.DataFormat.ASCii, data_length =␣
↪1)
```

Selects the format for numeric data transferred to and from the CMX500, for example query results.

param data_type

- ASCii: Numeric data is transferred as ASCII bytes. Floating point numbers are transferred in scientific E notation.
- REAL: Numeric data is transferred in a definite length block as IEEE floating point numbers (block data) .
- BINary | HEXadecimal | OCTal: Numeric data is transferred in binary, hexadecimal or octal format.

param data_length
The meaning depends on the DataType as listed below. A zero returned by a query means that the

default value is used.

- For ASCii: Decimal places of floating point numbers. That means, number of 'b' digits in the scientific notation a.bbbbbbE+ccc.Default: six decimal places
- For REAL: Length of floating point numbers in bits:32 bits = 4 bytes, format #14...64 bits = 8 bytes, format #18...Default: 64 bits
- For BINary, HEXadecimal, OCTal: Minimum number of digits. If the number is longer, more digits are used. If it is shorter, leading zeros are added.Default: 0, no leading zeros

6.12 Get

SCPI Command :

```
GET:XVALues
```

class GetCls

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

get_xvalues() → List[float]

```
# SCPI: GET:XVALues
value: List[float] = driver.get.get_xvalues()
```

No command help available

return
value: No help available

6.13 GlobalClearStatus

SCPI Command :

```
*GCLS
```

class GlobalClearStatusCls

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

set() → None

```
# SCPI: *GCLS
driver.globalClearStatus.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: *GCLS
driver.globalClearStatus.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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 GlobalWait

SCPI Command :

```
*GWAI
```

class GlobalWaitCls

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

set() → None

```
# SCPI: *GWAI
driver.globalWait.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: *GWAI
driver.globalWait.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.15 GotoLocal

SCPI Command :

```
*GTL
```

class GotoLocalCls

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

set() → None

```
# SCPI: *GTL
driver.gotoLocal.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: *GTL
driver.gotoLocal.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.16 HardCopy

SCPI Commands :

```
HCOPy:AREA
HCOPy:DATA
HCOPy:FILE
```

class HardCopyCls

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

get_area() → HardcopyArea

```
# SCPI: HCOPy:AREA
value: enums.HardcopyArea = driver.hardCopy.get_area()
```

No command help available

return
area: No help available

get_data() → bytes

```
# SCPI: HCOPy:DATA
value: bytes = driver.hardCopy.get_data()
```

No command help available

return
data: No help available

set_area(*area: HardcopyArea*) → None

```
# SCPI: HCOPy:AREA
driver.hardCopy.set_area(area = enums.HardcopyArea.AWINdow)
```

No command help available

param area
No help available

set_file(*filename: str*) → None

```
# SCPI: HCOPy:FILE
driver.hardCopy.set_file(filename = 'abc')
```

No command help available

param filename
No help available

Cloning the Group

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

Subgroups

6.16.1 Device

SCPI Command :

```
HCOPy:DEVice:FORMat
```

class DeviceCls

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

get_format_py() → ScreenshotFormat

```
# SCPI: HCOPy:DEVice:FORMat
value: enums.ScreenshotFormat = driver.hardCopy.device.get_format_py()
```

No command help available

return
file_formats: No help available

set_format_py(*file_formats: ScreenshotFormat*) → None

```
# SCPI: HCOPy:DEVice:FORMat
driver.hardCopy.device.set_format_py(file_formats = enums.ScreenshotFormat.BMP)
```

No command help available

param file_formats
No help available

6.16.2 Interior

SCPI Commands :

```
HCOPy:INTerior:DATA
HCOPy:INTerior:FILE
```

class InteriorCls

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

get_data() → bytes

```
# SCPI: HCOPy:INTerior:DATA
value: bytes = driver.hardCopy.interior.get_data()
```

No command help available

return
data: No help available

set_file(*filename: str*) → None

```
# SCPI: HCOPy:INTerior:FILE
driver.hardCopy.interior.set_file(filename = 'abc')
```

No command help available

param filename
No help available

6.17 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

6.17.1 Selftest

SCPI Command :

```
INIT:SELFtest
```

class SelftestCls

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

set(*opc_timeout_ms: int = -1*) → None

```
# SCPI: INIT:SELFtest
driver.init.selftest.set()
```

No command help available

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.18 Instrument

SCPI Command :

```
INSTrument:NSELect
```

class InstrumentCls

Instrument commands group definition. 8 total commands, 2 Subgroups, 1 group commands

get_nselect() → int

```
# SCPI: INSTrument:NSELect
value: int = driver.instrument.get_nselect()
```

No command help available

return
arg_0: No help available

set_nselect(*arg_0: int*) → None

```
# SCPI: INSTrument:NSELect
driver.instrument.set_nselect(arg_0 = 1)
```

No command help available

param arg_0
No help available

Cloning the Group

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

Subgroups

6.18.1 Display

SCPI Commands :

```
INSTrument:DISPlay:CAT
INSTrument:DISPlay:MODE
INSTrument:DISPlay:OPEN
INSTrument:DISPlay:CLOSe
INSTrument:DISPlay
```

class DisplayCls

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

close(*item: str*) → None

```
# SCPI: INSTrument:DISPlay:CLOSe
driver.instrument.display.close(item = 'abc')
```

No command help available

param item
No help available

get_cat() → str

```
# SCPI: INSTrument:DISPlay:CAT
value: str = driver.instrument.display.get_cat()
```

No command help available

return
list_py: No help available

get_mode() → DisplayMode

```
# SCPI: INSTrument:DISPlay:MODE
value: enums.DisplayMode = driver.instrument.display.get_mode()
```

No command help available

return
mode: No help available

get_value() → int

```
# SCPI: INSTrument:DISPlay
value: int = driver.instrument.display.get_value()
```

No command help available

return
instr: No help available

open(*item: str*) → None

```
# SCPI: INSTrument:DISPlay:OPEN
driver.instrument.display.open(item = 'abc')
```

No command help available

param item
No help available

set_mode(*mode: DisplayMode*) → None

```
# SCPI: INSTrument:DISPlay:MODE
driver.instrument.display.set_mode(mode = enums.DisplayMode.AUTomatic)
```

No command help available

param mode
No help available

set_value(*instr: int*) → None

```
# SCPI: INSTrument:DISPlay
driver.instrument.display.set_value(instr = 1)
```

No command help available

param instr
No help available

6.18.2 Select

SCPI Command :

```
INSTrument[:SELect]
```

class SelectCls

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

get_value() → str

```
# SCPI: INSTrument[:SELect]
value: str = driver.instrument.select.get_value()
```

No command help available

return
instrument: No help available

set_value(*instrument: str*) → None

```
# SCPI: INSTrument[:SELect]
driver.instrument.select.set_value(instrument = rawAbc)
```

No command help available

param instrument
No help available

Cloning the Group

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

Subgroups

6.18.2.1 Dstrategy

SCPI Command :

```
INSTrument[:SELect]:DSTRategy
```

class DstrategyCls

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

class DstrategyStruct

Response structure. Fields:

- Arg_0: enums.OperationMode: No parameter help available
- Arg_1: enums.DisplayStrategy: No parameter help available

get() → DstrategyStruct

```
# SCPI: INSTrument[:SELect]:DSTRategy
value: DstrategyStruct = driver.instrument.select.dstrategy.get()
```

No command help available

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

set(*arg_0: OperationMode*, *arg_1: DisplayStrategy = None*) → None

```
# SCPI: INSTrument[:SELect]:DSTRategy
driver.instrument.select.dstrategy.set(arg_0 = enums.OperationMode.LOCal, arg_1␣
↪= enums.DisplayStrategy.BYLayout)
```

No command help available

param arg_0
No help available

param arg_1
No help available

6.19 MacroCreate

SCPI Command :

```
*DMC
```

class MacroCreateCls

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

get(*label: str*) → str

```
# SCPI: *DMC
value: str = driver.macroCreate.get(label = 'abc')
```

Creates a macro. If the label exists already, the macro contents are overwritten. Macros are deleted when a remote connection is closed but can be saved to a macro file for later reuse, see method RsCMPX_Base.MassMemory.Store.Macro.set. Avoid using labels which are identical with supported remote control commands. In contrast to SCPI stipulations, remote commands have priority over macros.

param label
No help available

return
macro: No help available

set(*label: str*, *macro: str*) → None

```
# SCPI: *DMC
driver.macroCreate.set(label = 'abc', macro = 'abc')
```

Creates a macro. If the label exists already, the macro contents are overwritten. Macros are deleted when a remote connection is closed but can be saved to a macro file for later reuse, see method RsCMPX_Base.MassMemory.Store.Macro.set. Avoid using labels which are identical with supported remote control commands. In contrast to SCPI stipulations, remote commands have priority over macros.

param label
No help available

param macro
No help available

6.20 MassMemory

SCPI Commands :

```
MMEMory:COPY
MMEMory:DELete
MMEMory:DRIVes
MMEMory:MDIRectory
MMEMory:MOVE
MMEMory:MSIS
MMEMory:RDIRectory
MMEMory:SAV
MMEMory:RCL
MMEMory:ALIases
```

class MassMemoryCls

MassMemory commands group definition. 22 total commands, 6 Subgroups, 10 group commands

class AliasesStruct

Structure for reading output parameters. Fields:

- Alias: List[str]: No parameter help available
- Path: List[str]: No parameter help available

copy(*file_source: str*, *file_destination: str = None*) → None

```
# SCPI: MMEMory:COPY
driver.massMemory.copy(file_source = 'abc', file_destination = 'abc')
```

Copies an existing file. The target directory must exist.

param file_source
Name of the file to be copied. Wildcards ? and * are allowed if FileDestination contains a path without a filename.

param file_destination
Path and/or name of the new file If no file destination is specified, the source file is written to the current directory (see method RsCMPX_Base.MassMemory.CurrentDirectory.set) . Wildcards are not allowed.

delete(*filename: str*) → None

```
# SCPI: MMEMory:DELete
driver.massMemory.delete(filename = 'abc')
```

Deletes the specified files.

param filename
File to be deleted. The wildcards * and ? are allowed. Specifying a directory instead of a file is not allowed.

delete_directory(*directory_name: str*) → None

```
# SCPI: MMEMory:RDIRectory
driver.massMemory.delete_directory(directory_name = 'abc')
```

Removes an existing empty directory from the mass memory storage system.

param directory_name
Wildcards are not allowed.

get_aliases() → AliasesStruct

```
# SCPI: MMEMory:ALIases
value: AliasesStruct = driver.massMemory.get_aliases()
```

Returns the defined alias entries and the assigned directories. These settings are predefined and cannot be configured.

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

get_drives() → List[str]

```
# SCPI: MMEMory:DRIVes
value: List[str] = driver.massMemory.get_drives()
```

No command help available

return
drive: No help available

get_msis() → str

```
# SCPI: MMEMory:MSIS
value: str = driver.massMemory.get_msis()
```

No command help available

return
msus: No help available

make_directory(*directory_name: str*) → None

```
# SCPI: MMEMory:MDIRectory
driver.massMemory.make_directory(directory_name = 'abc')
```

Creates a directory. If necessary, an entire path consisting of several subdirectories is created.

param directory_name
Wildcards are not allowed.

move(*file_source: str*, *file_destination: str*) → None

```
# SCPI: MMEMory:MOVE
driver.massMemory.move(file_source = 'abc', file_destination = 'abc')
```

Moves or renames an existing object (file or directory) to a new location.

param file_source
Name of the object to be moved or renamed. Wildcards ? and * are only allowed for moving files without renaming.

param file_destination
New name and/or path of the object. Wildcards are not allowed. If a new object name without a path is specified, the object is renamed. If a new path without an object name

is specified, the object is moved to this path. If a new path and a new object name are specified, the object is moved to this path and renamed.

recall(*filename: str*, *msus: str = None*) → None

```
# SCPI: MMEMory:RCL
driver.massMemory.recall(filename = 'abc', msus = 'abc')
```

Restores the instrument settings from the specified file. This command has the same effect as the combination of method RsCMPX_Base.MassMemory.Load.State.set and *RCL.

param filename
No help available

param msus
No help available

save(*filename: str*, *msus: str = None*) → None

```
# SCPI: MMEMory:SAV
driver.massMemory.save(filename = 'abc', msus = 'abc')
```

Stores the current instrument settings to the specified file. This command has the same effect as the combination of *SAV and method RsCMPX_Base.MassMemory.Store.State.set.

param filename
No help available

param msus
No help available

set_msis(*msus: str*) → None

```
# SCPI: MMEMory:MSIS
driver.massMemory.set_msis(msus = 'abc')
```

No command help available

param msus
No help available

Cloning the Group

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

Subgroups

6.20.1 Attribute

SCPI Command :

```
MMEMory:ATTRibute
```

class AttributeCls

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

get(*path_name: str*) → List[str]

```
# SCPI: MMEMory:ATTRibute
value: List[str] = driver.massMemory.attribute.get(path_name = 'abc')
```

Sets or removes file attributes for files and directories.

param path_name
No help available

return
file_entry: Comma-separated list of strings. Information strings are returned for the directories '.' and '..', for files and for subdirectories. Each string has the format 'ObjectName,Attributes'.

set(*path_name: str*, *attributes: str*) → None

```
# SCPI: MMEMory:ATTRibute
driver.massMemory.attribute.set(path_name = 'abc', attributes = 'abc')
```

Sets or removes file attributes for files and directories.

param path_name
No help available

param attributes
No help available

6.20.2 Catalog

SCPI Command :

```
MMEMory:CATalog
```

class CatalogCls

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

class GetStruct

Response structure. Fields:

- Used_Memory: int: No parameter help available
- Free_Memory: int: No parameter help available
- File_Entry: List[str]: No parameter help available

get(*path_name: str = None*, *format_py: CatalogFormat = None*) → GetStruct

```
# SCPI: MMEMory:CATalog
value: GetStruct = driver.massMemory.catalog.get(path_name = 'abc', format_py =␣
↪enums.CatalogFormat.ALL)
```

Returns information on the contents of the current or of a specified directory.

param path_name
No help available

param format_py
No help available

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

Cloning the Group

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

Subgroups

6.20.2.1 Length

SCPI Command :

```
MMEMory:CATalog:LENGth
```

class LengthCls

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

get(*path_name: str = None*) → int

```
# SCPI: MMEMory:CATalog:LENGth
value: int = driver.massMemory.catalog.length.get(path_name = 'abc')
```

Returns the number of files and subdirectories of the current or of a specified directory. The number also considers the strings '.' and '..' so that it corresponds to the number of strings returned by the method RsCMPX_Base.MassMemory. **Catalog.get_** command after the initial numeric parameters.

param path_name
If the directory name is omitted, the command queries the contents of the current directory (see method RsCMPX_Base.MassMemory.CurrentDirectory.set) . If the wildcards ? or * are used, the number of files and subdirectories matching this pattern are returned.

return
count: No help available

6.20.3 CurrentDirectory

SCPI Command :

```
MMEMory:CDIRectory
```

class CurrentDirectoryCls

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

get(*directory_name: str = None*) → str

```
# SCPI: MMEMory:CDIRectory
value: str = driver.massMemory.currentDirectory.get(directory_name = 'abc')
```

Changes the current directory for mass memory storage. If <DirectoryName> is omitted, the current directory is set to '/'.

param directory_name
Wildcards are not allowed.

return
directory_name: Wildcards are not allowed.

set(*directory_name: str = None*) → None

```
# SCPI: MMEMory:CDIRectory
driver.massMemory.currentDirectory.set(directory_name = 'abc')
```

Changes the current directory for mass memory storage. If <DirectoryName> is omitted, the current directory is set to '/'.

param directory_name
Wildcards are not allowed.

6.20.4 Dcatalog

SCPI Command :

```
MMEMory:DCATalog
```

class DcatalogCls

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

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

```
# SCPI: MMEMory:DCATalog
value: List[str] = driver.massMemory.dcatalog.get(path_name = 'abc')
```

Returns the subdirectories of the current or of a specified directory.

param path_name
If this parameter is omitted, the command queries the contents of the current directory (see method RsCMPX_Base.MassMemory.CurrentDirectory.set) . If the wildcards ? or * are used, only the subdirectories matching this pattern are returned.

return
file_entry: Comma-separated list of strings with subdirectory names, one string per name

Cloning the Group

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

Subgroups

6.20.4.1 Length

SCPI Command :

```
MMEMory:DCATalog:LENGth
```

class LengthCls

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

get(*path_name: str = None*) → int

```
# SCPI: MMEMory:DCATalog:LENGth
value: int = driver.massMemory.dcatalog.length.get(path_name = 'abc')
```

Returns the number of subdirectories of the current or of a specified directory. The number also considers the strings '. ' and '..' so that it corresponds to the number of strings returned by the method RsCMPX_Base.MassMemory.Dcatalog. **get_** command.

param path_name
If the directory name is omitted, the command queries the contents of the current directory (see method RsCMPX_Base.MassMemory.CurrentDirectory.set) . If the wildcards ? or * are used, the number of subdirectories matching this pattern is returned.

return
file_entry_count: No help available

6.20.5 Load

class LoadCls

Load 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.massMemory.load.clone()
```

Subgroups

6.20.5.1 Item

SCPI Command :

```
MMEMory:LOAD:ITEM
```

class ItemCls

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

set(*item_path: str*, *filename: str*) → None

```
# SCPI: MMEMory:LOAD:ITEM
driver.massMemory.load.item.set(item_path = 'abc', filename = 'abc')
```

Executes a partial recall. That means, restores a selected part of a save file. You can restore all settings of a specific application instance. Or you can restore the list mode settings of a specific measurement application instance.

param item_path
No help available

param filename
No help available

6.20.5.2 Macro

SCPI Command :

```
MMEMory:LOAD:MACRo
```

class MacroCls

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

set(*label: str*, *filename: str*, *msus: str = None*) → None

```
# SCPI: MMEMory:LOAD:MACRo
driver.massMemory.load.macro.set(label = 'abc', filename = 'abc', msus = 'abc')
```

Creates a macro, reading the macro contents from a file. If the label exists already, the macro contents are overwritten. Avoid using labels which are identical with supported remote control commands. In contrast to SCPI stipulations, remote commands have priority over macros.

param label
No help available

param filename
No help available

param msus
No help available

6.20.5.3 State

SCPI Command :

```
MMEMory:LOAD:STATe
```

class StateCls

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

set(*sav_rcl_state_number: float*, *filename: str*, *msus: str = None*) → None

```
# SCPI: MMEMory:LOAD:STATe
driver.massMemory.load.state.set(sav_rcl_state_number = 1.0, filename = 'abc',␣
↪msus = 'abc')
```

Loads the instrument settings from the specified file to the specified internal memory. After the file has been loaded, the settings must be activated using a *RCL command. For more convenience, see method RsCMPX_Base.MassMemory.recall.

param sav_rcl_state_number
No help available

param filename
No help available

param msus
No help available

6.20.6 Store

class StoreCls

Store 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.massMemory.store.clone()
```

Subgroups

6.20.6.1 Item

SCPI Command :

```
MMEMory:STORe:ITEM
```

class ItemCls

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

set(*item_name: str*, *filename: str*) → None

```
# SCPI: MMEMory:STORe:ITEM
driver.massMemory.store.item.set(item_name = 'abc', filename = 'abc')
```

Executes a partial save, i.e. stores a part of the instrument settings to the specified file. You can store all settings of a specific application instance. Or you can store the list mode settings of a specific measurement application instance.

param item_name
Part to be saved. ItemName = Application[i][:MEV:LIST] For Application, see method RsCMPX_Base.MassMemory.Load.Item.set. i is the instance of the application. Omitting i stores instance 1. Appending :MEV:LIST stores only the list mode settings.

param filename
Path and filename of the target file. Wildcards are not allowed.

6.20.6.2 Macro

SCPI Command :

```
MMEMory:STORe:MACRo
```

class MacroCls

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

set(*label: str*, *filename: str*, *msus: str = None*) → None

```
# SCPI: MMEMory:STORe:MACRo
driver.massMemory.store.macro.set(label = 'abc', filename = 'abc', msus = 'abc')
```

Stores the contents of a macro to a file. If the file exists, it is overwritten. If the file does not exist, it is created.

param label
No help available

param filename
No help available

param msus
No help available

6.20.6.3 State

SCPI Command :

```
MMEMory:STORe:STATe
```

class StateCls

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

set(*sav_rcl_state_number: int*, *filename: str*, *msus: str = None*) → None

```
# SCPI: MMEMory:STORe:STATe
driver.massMemory.store.state.set(sav_rcl_state_number = 1, filename = 'abc',␣
↪msus = 'abc')
```

Stores the instrument settings from the specified internal memory to the specified file. To store the current instrument settings to a file, use first *SAV <MemoryNumber> to store the settings to the memory. Then use this command to store the settings from the memory to a file. For more convenience, see method RsCMPX_Base.MassMemory.save.

param sav_rcl_state_number
No help available

param filename
No help available

param msus
No help available

6.21 Modify

class ModifyCls

Modify 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.modify.clone()
```

Subgroups

6.21.1 System

class SystemCls

System 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.modify.system.clone()
```

Subgroups

6.21.1.1 Attenuation

class AttenuationCls

Attenuation 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.modify.system.attenuation.clone()
```

Subgroups

6.21.1.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.modify.system.attenuation.correctionTable.clone()
```

Subgroups

6.21.1.1.1.1 Globale

SCPI Command :

```
MODify:SYSTem:ATTenuation:CTABle:GLOBal
```

class GlobaleCls

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

set(*name: str*, *frequency: List[float]*, *attenuation: List[float]*) → None

```
# SCPI: MODify:SYSTem:ATTenuation:CTABle:GLOBal
driver.modify.system.attenuation.correctionTable.globale.set(name = 'abc',␣
↪frequency = [1.1, 2.2, 3.3], attenuation = [1.1, 2.2, 3.3])
```

Modifies existing entries of a global correction table. Specify at least one parameter pair <Frequency>, <Attenuation>. Entries with the specified frequencies must already exist. The attenuation values of these entries are overwritten.

param name
Name of the correction table

param frequency
No help available

param attenuation
No help available

6.21.1.1.1.2 Tenvironment

SCPI Command :

```
MODify:SYSTem:ATTenuation:CTABle[:TENVironment]
```

class TenvironmentCls

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

set(*name: str*, *frequency: List[float]*, *attenuation: List[float]*) → None

```
# SCPI: MODify:SYSTem:ATTenuation:CTABle[:TENVironment]
driver.modify.system.attenuation.correctionTable.tenvironment.set(name = 'abc',␣
↪frequency = [1.1, 2.2, 3.3], attenuation = [1.1, 2.2, 3.3])
```

Modifies existing entries of a channel-specific correction table. Specify at least one parameter pair <Frequency>, <Attenuation>. Entries with the specified frequencies must already exist. The attenuation values of these entries are overwritten.

param name
Name of the correction table

param frequency
No help available

param attenuation
No help available

6.22 Procedure

SCPI Command :

```
PROCedure:CMWD
```

class ProcedureCls

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

get_cmwd() → str

```
# SCPI: PROCedure:CMWD
value: str = driver.procedure.get_cmwd()
```

No command help available

return
command_string: No help available

set_cmwd(*command_string: str*) → None

```
# SCPI: PROCedure:CMWD
driver.procedure.set_cmwd(command_string = 'abc')
```

No command help available

param command_string
No help available

6.23 RecallState

SCPI Command :

```
*RCL
```

class RecallStateCls

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

set(*num: float*) → None

```
# SCPI: *RCL
driver.recallState.set(num = 1.0)
```

Loads the instrument settings from an intermediate memory identified by the specified number. The instrument settings can be stored to this memory using the command *SAV with the associated number. To load instrument settings from a file to the memory, see method RsCMPX_Base.MassMemory.Load.State.set. See also method RsCMPX_Base.MassMemory.recall.

param num
No help available

6.24 Remove

class RemoveCls

Remove 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.remove.clone()
```

Subgroups

6.24.1 System

class SystemCls

System 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.remove.system.clone()
```

Subgroups

6.24.1.1 Attenuation

class AttenuationCls

Attenuation 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.remove.system.attenuation.clone()
```

Subgroups

6.24.1.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.system.attenuation.correctionTable.clone()
```

Subgroups

6.24.1.1.1.1 Globale

SCPI Command :

```
REMove:SYSTem:ATTenuation:CTABle:GLOBal
```

class GlobaleCls

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

set(*name: str*, *intervall_start: float*, *intervall_end: float = None*) → None

```
# SCPI: REMove:SYSTem:ATTenuation:CTABle:GLOBal
driver.remove.system.attenuation.correctionTable.globale.set(name = 'abc',␣
↪intervall_start = 1.0, intervall_end = 1.0)
```

Removes entries from an existing global correction table. To remove a single entry, enter the frequency of the entry as <IntervallStart> and omit <IntervallEnd>. To remove all entries in a certain frequency range, enter the lower frequency as <IntervallStart> and the upper frequency as <IntervallEnd>.

param name
Name of the existing correction table.

param intervall_start
No help available

param intervall_end
No help available

6.24.1.1.1.2 Tenvironment

SCPI Command :

```
REMove:SYSTem:ATTenuation:CTABle[:TENVironment]
```

class TenvironmentCls

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

set(*name: str*, *intervall_start: float*, *intervall_end: float = None*) → None

```
# SCPI: REMove:SYSTem:ATTenuation:CTABle[:TENVironment]
driver.remove.system.attenuation.correctionTable.tenvironment.set(name = 'abc',␣
↪intervall_start = 1.0, intervall_end = 1.0)
```

Removes entries from an existing channel-specific correction table. To remove a single entry, enter the frequency of the entry as <IntervallStart> and omit <IntervallEnd>. To remove all entries in a certain frequency range, enter the lower frequency as <IntervallStart> and the upper frequency as <IntervallEnd>.

param name
Name of the existing correction table.

param intervall_start
No help available

param intervall_end
No help available

6.24.2 Tenvironment

class TenvironmentCls

Tenvironment 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.remove.tenvironment.clone()
```

Subgroups

6.24.2.1 Spath

class SpathCls

Spath 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.remove.tenvironment.spath.clone()
```

Subgroups

6.24.2.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.tenvironment.spath.correctionTable.clone()
```

Subgroups

6.24.2.1.1.1 Rx

SCPI Command :

```
REMove:TENVironment:SPATh:CTABle:RX
```

class RxCls

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

set(*name_signal_path: str*, *correction_table: List[str] = None*) → None

```
# SCPI: REMove:TENVironment:SPATh:CTABle:RX
driver.remove.tenvironment.spath.correctionTable.rx.set(name_signal_path = 'abc
↪', correction_table = ['abc1', 'abc2', 'abc3'])
```

Removes assigned correction tables from the TX direction or RX direction of a connection.

param name_signal_path
Name of the connection

param correction_table
The name of the correction table to be removed. If you omit the parameter, all correction tables are removed. You can specify several table names as comma-separated list of strings.

6.24.2.1.1.2 Tx

SCPI Command :

```
REMove:TENVironment:SPATh:CTABle:TX
```

class TxCls

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

set(*name_signal_path: str*, *correction_table: List[str] = None*) → None

```
# SCPI: REMove:TENVironment:SPATh:CTABle:TX
driver.remove.tenvironment.spath.correctionTable.tx.set(name_signal_path = 'abc
↪', correction_table = ['abc1', 'abc2', 'abc3'])
```

Removes assigned correction tables from the TX direction or RX direction of a connection.

param name_signal_path
Name of the connection

param correction_table
The name of the correction table to be removed. If you omit the parameter, all correction tables are removed. You can specify several table names as comma-separated list of strings.

6.25 Route

class RouteCls

Route commands group definition. 29 total commands, 14 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.25.1 Bluetooth

class BluetoothCls

Bluetooth 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.route.bluetooth.clone()
```

Subgroups

6.25.1.1 Measurement

class MeasurementCls

Measurement 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.route.bluetooth.measurement.clone()
```

Subgroups

6.25.1.1.1 Spath

SCPI Commands :

```
ROUTe:BLUetooth:MEASurement<Instance>:SPATh:COUNt
ROUTe:BLUetooth:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:BLUetooth:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.bluetooth.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:BLUetooth:MEASurement<Instance>:SPATh
value: List[str] = driver.route.bluetooth.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:BLUetooth:MEASurement<Instance>:SPATh
driver.route.bluetooth.measurement.spath.set_value(signal_path = ['abc1', 'abc2
↪', 'abc3'])
```

No command help available

param signal_path
No help available

6.25.2 Cdma

class CdmaCls

Cdma 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.route.cdma.clone()
```

Subgroups

6.25.2.1 Measurement

SCPI Command :

```
ROUTe:CDMA:MEASurement<Instance>:SPATh
```

class MeasurementCls

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

get_spath() → List[str]

```
# SCPI: ROUTe:CDMA:MEASurement<Instance>:SPATh
value: List[str] = driver.route.cdma.measurement.get_spath()
```

No command help available

return
signal_path: No help available

set_spath(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:CDMA:MEASurement<Instance>:SPATh
driver.route.cdma.measurement.set_spath(signal_path = ['abc1', 'abc2', 'abc3'])
```

No command help available

param signal_path
No help available

6.25.3 Gprf

class GprfCls

Gprf 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.route.gprf.clone()
```

Subgroups

6.25.3.1 Generator

class GeneratorCls

Generator 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.route.gprf.generator.clone()
```

Subgroups

6.25.3.1.1 Spath

SCPI Commands :

```
ROUTe:GPRF:GENerator<Instance>:SPATh:COUNt
ROUTe:GPRF:GENerator<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:GPRF:GENerator<Instance>:SPATh:COUNt
value: int = driver.route.gprf.generator.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:GPRF:GENerator<Instance>:SPATh
value: List[str] = driver.route.gprf.generator.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:GPRF:GENerator<Instance>:SPATh
driver.route.gprf.generator.spath.set_value(signal_path = ['abc1', 'abc2', 'abc3
↪'])
```

No command help available

param signal_path
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.route.gprf.generator.spath.clone()
```

Subgroups

6.25.3.1.1.1 Group

SCPI Command :

```
ROUTe:GPRF:GENerator<Instance>:SPATh:GROup
```

class GroupCls

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

get(*connector_name: str*) → str

```
# SCPI: ROUTe:GPRF:GENerator<Instance>:SPATh:GROup
value: str = driver.route.gprf.generator.spath.group.get(connector_name = 'abc')
```

No command help available

param connector_name
No help available

return
signal_path: No help available

set(*connector_name: str*, *signal_path: str*) → None

```
# SCPI: ROUTe:GPRF:GENerator<Instance>:SPATh:GROup
driver.route.gprf.generator.spath.group.set(connector_name = 'abc', signal_path
↪= 'abc')
```

No command help available

param connector_name
No help available

param signal_path
No help available

6.25.3.2 Measurement

class MeasurementCls

Measurement 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.route.gprf.measurement.clone()
```

Subgroups

6.25.3.2.1 Spath

SCPI Commands :

```
ROUTe:GPRF:MEASurement<Instance>:SPATh:COUNt
ROUTe:GPRF:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:GPRF:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.gprf.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:GPRF:MEASurement<Instance>:SPATh
value: List[str] = driver.route.gprf.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:GPRF:MEASurement<Instance>:SPATh
driver.route.gprf.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.4 Gsm

class GsmCls

Gsm 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.route.gsm.clone()
```

Subgroups

6.25.4.1 Measurement

class MeasurementCls

Measurement 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.route.gsm.measurement.clone()
```

Subgroups

6.25.4.1.1 Spath

SCPI Commands :

```
ROUTe:GSM:MEASurement<Instance>:SPATh:COUNt
ROUTe:GSM:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.gsm.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SPATh
value: List[str] = driver.route.gsm.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:GSM:MEASurement<Instance>:SPATh
driver.route.gsm.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.5 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.route.lte.clone()
```

Subgroups

6.25.5.1 Measurement

class MeasurementCls

Measurement 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.route.lte.measurement.clone()
```

Subgroups

6.25.5.1.1 Spath

SCPI Commands :

```
ROUTe:LTE:MEASurement<Instance>:SPATh:COUNt
ROUTe:LTE:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.lte.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SPATh
value: List[str] = driver.route.lte.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:LTE:MEASurement<Instance>:SPATh
driver.route.lte.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.6 LteDl

class LteDlCls

LteDl 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.route.lteDl.clone()
```

Subgroups

6.25.6.1 Measurement

class MeasurementCls

Measurement 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.route.lteDl.measurement.clone()
```

Subgroups

6.25.6.1.1 Spath

SCPI Commands :

```
ROUTe:LTEDl:MEASurement<Instance>:SPATh:COUNt
ROUTe:LTEDl:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:LTEDl:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.lteDl.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:LTEDl:MEASurement<Instance>:SPATh
value: List[str] = driver.route.lteDl.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:LTEDl:MEASurement<Instance>:SPATh
driver.route.lteDl.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.7 Niot

class NiotCls

Niot 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.route.niot.clone()
```

Subgroups

6.25.7.1 Measurement

SCPI Command :

```
ROUTe:NIOT:MEASurement<Instance>:SPATh
```

class MeasurementCls

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

get_spath() → List[str]

```
# SCPI: ROUTe:NIOT:MEASurement<Instance>:SPATh
value: List[str] = driver.route.niot.measurement.get_spath()
```

No command help available

return
signal_path: No help available

set_spath(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:NIOT:MEASurement<Instance>:SPATh
driver.route.niot.measurement.set_spath(signal_path = ['abc1', 'abc2', 'abc3'])
```

No command help available

param signal_path
No help available

6.25.8 NrDl

class NrDlCls

NrDl 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.route.nrDl.clone()
```

Subgroups

6.25.8.1 Measurement

class MeasurementCls

Measurement 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.route.nrDl.measurement.clone()
```

Subgroups

6.25.8.1.1 Spath

SCPI Commands :

```
ROUTe:NRDL:MEASurement<Instance>:SPATh:COUNt
ROUTe:NRDL:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:NRDL:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.nrDl.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:NRDL:MEASurement<Instance>:SPATh
value: List[str] = driver.route.nrDl.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:NRDL:MEASurement<Instance>:SPATh
driver.route.nrDl.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.9 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.route.nrMmw.clone()
```

Subgroups

6.25.9.1 Measurement

class MeasurementCls

Measurement 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.route.nrMmw.measurement.clone()
```

Subgroups

6.25.9.1.1 Spath

SCPI Commands :

```
ROUTe:NRMMw:MEASurement<Instance>:SPATh:COUNt
ROUTe:NRMMw:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:NRMMw:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.nrMmw.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:NRMMw:MEASurement<Instance>:SPATh
value: List[str] = driver.route.nrMmw.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:NRMMw:MEASurement<Instance>:SPATh
driver.route.nrMmw.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.10 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.route.nrSub.clone()
```

Subgroups

6.25.10.1 Measurement

class MeasurementCls

Measurement 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.route.nrSub.measurement.clone()
```

Subgroups

6.25.10.1.1 Spath

SCPI Commands :

```
ROUTe:NRSub:MEASurement<Instance>:SPATh:COUNt
ROUTe:NRSub:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:NRSub:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.nrSub.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:NRSub:MEASurement<Instance>:SPATh
value: List[str] = driver.route.nrSub.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:NRSub:MEASurement<Instance>:SPATh
driver.route.nrSub.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.11 Uwb

class UwbCls

Uwb 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.route.uwb.clone()
```

Subgroups

6.25.11.1 Measurement

class MeasurementCls

Measurement 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.route.uwb.measurement.clone()
```

Subgroups

6.25.11.1.1 Spath

SCPI Commands :

```
ROUTe:UWB:MEASurement<Instance>:SPATh:COUNt
ROUTe:UWB:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:UWB:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.uwb.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:UWB:MEASurement<Instance>:SPATh
value: List[str] = driver.route.uwb.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:UWB:MEASurement<Instance>:SPATh
driver.route.uwb.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.12 Wcdma

class WcdmaCls

Wcdma 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.route.wcdma.clone()
```

Subgroups

6.25.12.1 Measurement

class MeasurementCls

Measurement 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.route.wcdma.measurement.clone()
```

Subgroups

6.25.12.1.1 Spath

SCPI Commands :

```
ROUTe:WCDMa:MEASurement<Instance>:SPATh:COUNt
ROUTe:WCDMa:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:WCDMa:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.wcdma.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:WCDMa:MEASurement<Instance>:SPATh
value: List[str] = driver.route.wcdma.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:WCDMa:MEASurement<Instance>:SPATh
driver.route.wcdma.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.13 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.route.wlan.clone()
```

Subgroups

6.25.13.1 Measurement

class MeasurementCls

Measurement 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.route.wlan.measurement.clone()
```

Subgroups

6.25.13.1.1 Spath

SCPI Commands :

```
ROUTe:WLAN:MEASurement<Instance>:SPATh:COUNt
ROUTe:WLAN:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.wlan.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SPATh
value: List[str] = driver.route.wlan.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:WLAN:MEASurement<Instance>:SPATh
driver.route.wlan.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.25.14 Wpan

class WpanCls

Wpan 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.route.wpan.clone()
```

Subgroups

6.25.14.1 Measurement

class MeasurementCls

Measurement 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.route.wpan.measurement.clone()
```

Subgroups

6.25.14.1.1 Spath

SCPI Commands :

```
ROUTe:WPAN:MEASurement<Instance>:SPATh:COUNt
ROUTe:WPAN:MEASurement<Instance>:SPATh
```

class SpathCls

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

get_count() → int

```
# SCPI: ROUTe:WPAN:MEASurement<Instance>:SPATh:COUNt
value: int = driver.route.wpan.measurement.spath.get_count()
```

No command help available

return
signal_path_count: No help available

get_value() → List[str]

```
# SCPI: ROUTe:WPAN:MEASurement<Instance>:SPATh
value: List[str] = driver.route.wpan.measurement.spath.get_value()
```

No command help available

return
signal_path: No help available

set_value(*signal_path: List[str]*) → None

```
# SCPI: ROUTe:WPAN:MEASurement<Instance>:SPATh
driver.route.wpan.measurement.spath.set_value(signal_path = ['abc1', 'abc2',
↪'abc3'])
```

No command help available

param signal_path
No help available

6.26 SaveState

SCPI Command :

```
*SAV
```

class SaveStateCls

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

set(*num: float*) → None

```
# SCPI: *SAV
driver.saveState.set(num = 1.0)
```

Stores the current instrument settings under the specified number in an intermediate memory. The settings can be restored, using the command *RCL with the associated number. To save the stored instrument settings to a file, see method RsCMPX_Base.MassMemory.Store.State.set. See also method RsCMPX_Base.MassMemory.save.

param num
No help available

6.27 Selftest

SCPI Commands :

```
ABORt:SELFtest
STOP:SELFtest
FETCh:SELFtest
READ:SELFtest
```

class SelftestCls

Selftest commands group definition. 12 total commands, 4 Subgroups, 4 group commands

abort(*opc_timeout_ms: int = -1*) → None

```
# SCPI: ABORt:SELFtest
driver.selftest.abort()
```

No command help available

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

fetch(*filter_py: str = None*) → List[str]

```
# SCPI: FETCh:SELFtest
value: List[str] = driver.selftest.fetch(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

read(*filter_py: str = None*) → List[str]

```
# SCPI: READ:SELFtest
value: List[str] = driver.selftest.read(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

stop(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STOP:SELFtest
driver.selftest.stop()
```

No command help available

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

Cloning the Group

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

Subgroups

6.27.1 Failed

SCPI Commands :

```
FETCh:SELFtest:FAILed
READ:SELFtest:FAILed
```

class FailedCls

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

fetch(*filter_py: str = None*) → List[str]

```
# SCPI: FETCh:SELFtest:FAILed
value: List[str] = driver.selftest.failed.fetch(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

read(*filter_py: str = None*) → List[str]

```
# SCPI: READ:SELFtest:FAILed
value: List[str] = driver.selftest.failed.read(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

6.27.2 Passed

SCPI Commands :

```
FETCh:SELFtest:PASSed
READ:SELFtest:PASSed
```

class PassedCls

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

fetch(*filter_py: str = None*) → List[str]

```
# SCPI: FETCh:SELFtest:PASSed
value: List[str] = driver.selftest.passed.fetch(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

read(*filter_py: str = None*) → List[str]

```
# SCPI: READ:SELFtest:PASSed
value: List[str] = driver.selftest.passed.read(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

6.27.3 Skipped

SCPI Commands :

```
FETCh:SELFtest:SKIPped
READ:SELFtest:SKIPped
```

class SkippedCls

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

fetch(*filter_py: str = None*) → List[str]

```
# SCPI: FETCh:SELFtest:SKIPped
value: List[str] = driver.selftest.skipped.fetch(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

read(*filter_py: str = None*) → List[str]

```
# SCPI: READ:SELFtest:SKIPped
value: List[str] = driver.selftest.skipped.read(filter_py = 'abc')
```

No command help available

Suppressed linked return values: reliability

param filter_py
No help available

return
value: No help available

6.27.4 State

SCPI Command :

```
FETCh:SELFtest:STATe
```

class StateCls

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

fetch() → ResourceState

```
# SCPI: FETCh:SELFtest:STATe
value: enums.ResourceState = driver.selftest.state.fetch()
```

No command help available

return
meas_status: No help available

Cloning the Group

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

Subgroups

6.27.4.1 All

SCPI Command :

```
FETCh:SELFtest:STATe:ALL
```

class AllCls

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

fetch() → List[ResourceState]

```
# SCPI: FETCh:SELFtest:STATe:ALL
value: List[enums.ResourceState] = driver.selftest.state.all.fetch()
```

No command help available

return
meas_status: No help available

6.28 Sense

class SenseCls

Sense commands group definition. 15 total commands, 3 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.28.1 Base

class BaseCls

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

Cloning the Group

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

Subgroups

6.28.1.1 IpSet

class IpSetCls

IpSet 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.sense.base.ipSet.clone()
```

Subgroups

6.28.1.1.1 Snode

SCPI Commands :

```
SENSe:BASE:IPSet:SNODe:NNAMe
SENSe:BASE:IPSet:SNODe:NTYPe
SENSe:BASE:IPSet:SNODe:NSEGment
```

class SnodeCls

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

class NsegmentStruct

Structure for reading output parameters. Fields:

- Selected_Segment: enums.Segment: No parameter help available
- Ip_Address: str: No parameter help available
- Subnet_Mask: str: No parameter help available

get_nname() → str

```
# SCPI: SENSe:BASE:IPSet:SNODe:NNAMe
value: str = driver.sense.base.ipSet.snode.get_nname()
```

No command help available

return
name: No help available

get_nsegment() → NsegmentStruct

```
# SCPI: SENSe:BASE:IPSet:SNODe:NSEGment
value: NsegmentStruct = driver.sense.base.ipSet.snode.get_nsegment()
```

No command help available

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

get_ntype() → str

```
# SCPI: SENSe:BASE:IPSet:SNODe:NTYPe
value: str = driver.sense.base.ipSet.snode.get_ntype()
```

No command help available

return
type_py: No help available

6.28.1.1.2 SubMonitor

SCPI Commands :

```
SENSe:BASE:IPSet:SMONitor:NAME
SENSe:BASE:IPSet:SMONitor:TYPE
SENSe:BASE:IPSet:SMONitor:ID
SENSe:BASE:IPSet:SMONitor:DESCription
```

class SubMonitorCls

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

get_description() → List[str]

```
# SCPI: SENSe:BASE:IPSet:SMONitor:DESCription
value: List[str] = driver.sense.base.ipSet.subMonitor.get_description()
```

No command help available

return
descriptions: No help available

get_id() → List[int]

```
# SCPI: SENSe:BASE:IPSet:SMONitor:ID
value: List[int] = driver.sense.base.ipSet.subMonitor.get_id()
```

No command help available

return
ids: No help available

get_name() → List[str]

```
# SCPI: SENSe:BASE:IPSet:SMONitor:NAME
value: List[str] = driver.sense.base.ipSet.subMonitor.get_name()
```

No command help available

return
names: No help available

get_type_py() → List[str]

```
# SCPI: SENSe:BASE:IPSet:SMONitor:TYPE
value: List[str] = driver.sense.base.ipSet.subMonitor.get_type_py()
```

No command help available

return
types: No help available

6.28.1.2 Reference

class ReferenceCls

Reference 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.base.reference.clone()
```

Subgroups

6.28.1.2.1 Frequency

SCPI Command :

```
SENSe:BASE:REFerence:FREQuency:LOCKed
```

class FrequencyCls

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

get_locked() → bool

```
# SCPI: SENSe:BASE:REFerence:FREQuency:LOCKed
value: bool = driver.sense.base.reference.frequency.get_locked()
```

Queries whether the reference frequency is locked or not.

return
lock: 1: The frequency is locked. 0: The frequency is not locked.

6.28.1.3 Temperature

SCPI Command :

```
SENSe:BASE:TEMPerature:ENVironment
```

class TemperatureCls

Temperature commands group definition. 5 total commands, 2 Subgroups, 1 group commands

get_environment() → float

```
# SCPI: SENSe:BASE:TEMPerature:ENVironment
value: float = driver.sense.base.temperature.get_environment()
```

No command help available

return
temperature: No help available

Cloning the Group

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

Subgroups

6.28.1.3.1 Exceeded

SCPI Commands :

```
SENSe:BASE:TEMPerature:EXCeeded:LIST
SENSe:BASE:TEMPerature:EXCeeded
```

class ExceededCls

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

class ListPyStruct

Structure for reading output parameters. Fields:

- Meas_Point: List[str]: No parameter help available
- Current_Temp: List[float]: No parameter help available
- Max_Temp: List[float]: No parameter help available

get_list_py() → ListPyStruct

```
# SCPI: SENSe:BASE:TEMPerature:EXCeeded:LIST
value: ListPyStruct = driver.sense.base.temperature.exceeded.get_list_py()
```

No command help available

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

get_value() → bool

```
# SCPI: SENSe:BASE:TEMPerature:EXCeeded
value: bool = driver.sense.base.temperature.exceeded.get_value()
```

No command help available

return
exceed: No help available

6.28.1.3.2 Operating

SCPI Command :

```
SENSe:BASE:TEMPerature:OPERating:INTernal
```

class OperatingCls

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

get_internal() → float

```
# SCPI: SENSe:BASE:TEMPerature:OPERating:INTernal
value: float = driver.sense.base.temperature.operating.get_internal()
```

No command help available

return
temperature: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.base.temperature.operating.clone()
```

Subgroups

6.28.1.3.2.1 Ambient

SCPI Command :

```
SENSe:BASE:TEMPerature:OPERating:AMBient
```

class AmbientCls

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

class GetStruct

Response structure. Fields:

- Temperature: float: Temperature in degree Celsius
- Timestamp: str: Uptime of the session, as 'days-hours-minutes-seconds'
- Box: str: Identifies the instrument in a setup with several connected instruments (for future use)

get(*all_py: All = None*) → GetStruct

```
# SCPI: SENSe:BASE:TEMPerature:OPERating:AMBient
value: GetStruct = driver.sense.base.temperature.operating.ambient.get(all_py =␣
↪enums.All.ALL)
```

Queries the ambient temperature measured via a sensor in the instrument. Ambient temperatures reported by self-tests are also measured via this sensor.

param all_py
For future use

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

6.28.2 FirmwareUpdate

SCPI Command :

```
SENSe:FWUPdate:INFO
```

class FirmwareUpdateCls

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

get_info() → str

```
# SCPI: SENSe:FWUPdate:INFO
value: str = driver.sense.firmwareUpdate.get_info()
```

No command help available

return
info: No help available

6.28.3 Selftest

class SelftestCls

Selftest 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.selftest.clone()
```

Subgroups

6.28.3.1 State

SCPI Command :

```
SENSe:SELFtest:STATe:SUM
```

class StateCls

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

get_sum() → SelftestSumState

```
# SCPI: SENSe:SELFtest:STATe:SUM
value: enums.SelftestSumState = driver.sense.selftest.state.get_sum()
```

No command help available

return
sum_state: No help available

6.29 Source

class SourceCls

Source 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.clone()
```

Subgroups

6.29.1 Base

class BaseCls

Base 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.base.clone()
```

Subgroups

6.29.1.1 Adjustment

class AdjustmentCls

Adjustment 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.base.adjustment.clone()
```

Subgroups

6.29.1.1.1 State

SCPI Command :

```
SOURce:BASE:ADJustment:STATe
```

class StateCls

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

get() → BaseAdjState

```
# SCPI: SOURce:BASE:ADJustment:STATe
value: enums.BaseAdjState = driver.source.base.adjustment.state.get()
```

No command help available

return
state: No help available

set(*control: bool*) → None

```
# SCPI: SOURce:BASE:ADJustment:STATe
driver.source.base.adjustment.state.set(control = False)
```

No command help available

param control
No help available

6.30 Status

SCPI Command :

```
STATus:PRESet
```

class StatusCls

Status commands group definition. 37 total commands, 7 Subgroups, 1 group commands

preset() → None

```
# SCPI: STATus:PRESet
driver.status.preset()
```

No command help available

preset_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: STATus:PRESet
driver.status.preset_with_opc()
```

No command help available

Same as preset, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.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.status.clone()
```

Subgroups

6.30.1 Condition

class ConditionCls

Condition 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.status.condition.clone()
```

Subgroups

6.30.1.1 Bits

class BitsCls

Bits 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.status.condition.bits.clone()
```

Subgroups

6.30.1.1.1 All

SCPI Command :

```
STATus:CONDition:BITS:ALL
```

class AllCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → List[str]

```
# SCPI: STATus:CONDition:BITS:ALL
value: List[str] = driver.status.condition.bits.all.get(filter_py = 'abc', mode␣
↪= enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bit: No help available

6.30.1.1.2 Cataloge

SCPI Command :

```
STATus:CONDition:BITS:CATaloge
```

class CatalogeCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → List[str]

```
# SCPI: STATus:CONDition:BITS:CATaloge
value: List[str] = driver.status.condition.bits.cataloge.get(filter_py = 'abc',␣
↪mode = enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bit: No help available

6.30.1.1.3 Count

SCPI Command :

```
STATus:CONDition:BITS:COUNt
```

class CountCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → int

```
# SCPI: STATus:CONDition:BITS:COUNt
value: int = driver.status.condition.bits.count.get(filter_py = 'abc', mode =␣
↪enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
count: No help available

6.30.2 Event

class EventCls

Event commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.30.2.1 Bits

SCPI Command :

```
STATus:EVENt:BITS:CLEar
```

class BitsCls

Bits commands group definition. 4 total commands, 3 Subgroups, 1 group commands

clear(*filter_py: str = None*, *mode: ExpressionMode = None*) → None

```
# SCPI: STATus:EVENt:BITS:CLEar
driver.status.event.bits.clear(filter_py = 'abc', mode = enums.ExpressionMode.
↪REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

Cloning the Group

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

Subgroups

6.30.2.1.1 All

SCPI Command :

```
STATus:EVENt:BITS:ALL
```

class AllCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → List[str]

```
# SCPI: STATus:EVENt:BITS:ALL
value: List[str] = driver.status.event.bits.all.get(filter_py = 'abc', mode =␣
↪enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bit: No help available

6.30.2.1.2 Count

SCPI Command :

```
STATus:EVENt:BITS:COUNt
```

class CountCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → int

```
# SCPI: STATus:EVENt:BITS:COUNt
value: int = driver.status.event.bits.count.get(filter_py = 'abc', mode = enums.
↪ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
count: No help available

6.30.2.1.3 Next

SCPI Command :

```
STATus:EVENt:BITS:NEXT
```

class NextCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:EVENt:BITS:NEXT
value: str = driver.status.event.bits.next.get(filter_py = 'abc', mode = enums.
↪ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bit: No help available

6.30.3 Generator

class GeneratorCls

Generator 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.status.generator.clone()
```

Subgroups

6.30.3.1 Condition

class ConditionCls

Condition 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.status.generator.condition.clone()
```

Subgroups

6.30.3.1.1 Off

SCPI Command :

```
STATus:GENerator:CONDition:OFF
```

class OffCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:GENerator:CONDition:OFF
value: str = driver.status.generator.condition.off.get(filter_py = 'abc', mode␣
↪= enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.3.1.2 On

SCPI Command :

```
STATus:GENerator:CONDition:ON
```

class OnCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:GENerator:CONDition:ON
value: str = driver.status.generator.condition.on.get(filter_py = 'abc', mode =␣
↪enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.3.1.3 Pending

SCPI Command :

```
STATus:GENerator:CONDition:PENDing
```

class PendingCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:GENerator:CONDition:PENDing
value: str = driver.status.generator.condition.pending.get(filter_py = 'abc',␣
↪mode = enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.4 Measurement

class MeasurementCls

Measurement 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.status.measurement.clone()
```

Subgroups

6.30.4.1 Condition

class ConditionCls

Condition 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.status.measurement.condition.clone()
```

Subgroups

6.30.4.1.1 Off

SCPI Command :

```
STATus:MEASurement:CONDition:OFF
```

class OffCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:MEASurement:CONDition:OFF
value: str = driver.status.measurement.condition.off.get(filter_py = 'abc',␣
↪mode = enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.4.1.2 Qued

SCPI Command :

```
STATus:MEASurement:CONDition:QUED
```

class QuedCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:MEASurement:CONDition:QUED
value: str = driver.status.measurement.condition.qued.get(filter_py = 'abc',␣
↪mode = enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.4.1.3 Rdy

SCPI Command :

```
STATus:MEASurement:CONDition:RDY
```

class RdyCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:MEASurement:CONDition:RDY
value: str = driver.status.measurement.condition.rdy.get(filter_py = 'abc',␣
↪mode = enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.4.1.4 Run

SCPI Command :

```
STATus:MEASurement:CONDition:RUN
```

class RunCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:MEASurement:CONDition:RUN
value: str = driver.status.measurement.condition.run.get(filter_py = 'abc',␣
↪mode = enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.4.1.5 SdReached

SCPI Command :

```
STATus:MEASurement:CONDition:SDReached
```

class SdReachedCls

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

get(*filter_py: str = None*, *mode: ExpressionMode = None*) → str

```
# SCPI: STATus:MEASurement:CONDition:SDReached
value: str = driver.status.measurement.condition.sdReached.get(filter_py = 'abc
↪', mode = enums.ExpressionMode.REGex)
```

No command help available

param filter_py
No help available

param mode
No help available

return
bitname: No help available

6.30.5 Operation

SCPI Commands :

```
STATus:OPERation[:EVENt]
STATus:OPERation:CONDition
STATus:OPERation:ENABle
STATus:OPERation:PTRansition
STATus:OPERation:NTRansition
```

class OperationCls

Operation commands group definition. 10 total commands, 1 Subgroups, 5 group commands

get_condition() → int

```
# SCPI: STATus:OPERation:CONDition
value: int = driver.status.operation.get_condition()
```

No command help available

return
register_value: No help available

get_enable() → int

```
# SCPI: STATus:OPERation:ENABle
value: int = driver.status.operation.get_enable()
```

No command help available

return
register_value: No help available

get_event() → int

```
# SCPI: STATus:OPERation[:EVENt]
value: int = driver.status.operation.get_event()
```

No command help available

return
register_value: No help available

get_ntransition() → int

```
# SCPI: STATus:OPERation:NTRansition
value: int = driver.status.operation.get_ntransition()
```

No command help available

return
register_value: No help available

get_ptransition() → int

```
# SCPI: STATus:OPERation:PTRansition
value: int = driver.status.operation.get_ptransition()
```

No command help available

return
register_value: No help available

set_enable(*register_value: int*) → None

```
# SCPI: STATus:OPERation:ENABle
driver.status.operation.set_enable(register_value = 1)
```

No command help available

param register_value
No help available

set_ntransition(*register_value: int*) → None

```
# SCPI: STATus:OPERation:NTRansition
driver.status.operation.set_ntransition(register_value = 1)
```

No command help available

param register_value
No help available

set_ptransition(*register_value: int*) → None

```
# SCPI: STATus:OPERation:PTRansition
driver.status.operation.set_ptransition(register_value = 1)
```

No command help available

param register_value
No help available

Cloning the Group

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

Subgroups

6.30.5.1 Bit<BitNr>

RepCap Settings

```
# Range: Nr8 .. Nr12
rc = driver.status.operation.bit.repcap_bitNr_get()
driver.status.operation.bit.repcap_bitNr_set(repcap.BitNr.Nr8)
```

class BitCls

Bit commands group definition. 5 total commands, 5 Subgroups, 0 group commands Repeated Capability: BitNr, default value after init: BitNr.Nr8

Cloning the Group

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

Subgroups

6.30.5.1.1 Condition

SCPI Command :

```
STATus:OPERation:BIT<bitno>:CONDition
```

class ConditionCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:OPERation:BIT<bitno>:CONDition
value: bool = driver.status.operation.bit.condition.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

6.30.5.1.2 Enable

SCPI Command :

```
STATus:OPERation:BIT<bitno>:ENABle
```

class EnableCls

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

get(*bitNr=BitNr.Default*) → float

```
# SCPI: STATus:OPERation:BIT<bitno>:ENABle
value: float = driver.status.operation.bit.enable.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

set(*register_bit: float*, *bitNr=BitNr.Default*) → None

```
# SCPI: STATus:OPERation:BIT<bitno>:ENABle
driver.status.operation.bit.enable.set(register_bit = 1.0, bitNr = repcap.BitNr.
↪Default)
```

No command help available

param register_bit
No help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

6.30.5.1.3 Event

SCPI Command :

```
STATus:OPERation:BIT<bitno>[:EVENt]
```

class EventCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:OPERation:BIT<bitno>[:EVENt]
value: bool = driver.status.operation.bit.event.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

6.30.5.1.4 Ntransition

SCPI Command :

```
STATus:OPERation:BIT<bitno>:NTRansition
```

class NtransitionCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:OPERation:BIT<bitno>:NTRansition
value: bool = driver.status.operation.bit.ntransition.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

set(*register_bit: bool*, *bitNr=BitNr.Default*) → None

```
# SCPI: STATus:OPERation:BIT<bitno>:NTRansition
driver.status.operation.bit.ntransition.set(register_bit = False, bitNr =␣
↪repcap.BitNr.Default)
```

No command help available

param register_bit
No help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

6.30.5.1.5 Ptransition

SCPI Command :

```
STATus:OPERation:BIT<bitno>:PTRansition
```

class PtransitionCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:OPERation:BIT<bitno>:PTRansition
value: bool = driver.status.operation.bit.ptransition.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

set(*register_bit: bool*, *bitNr=BitNr.Default*) → None

```
# SCPI: STATus:OPERation:BIT<bitno>:PTRansition
driver.status.operation.bit.ptransition.set(register_bit = False, bitNr =␣
↪repcap.BitNr.Default)
```

No command help available

param register_bit
No help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

6.30.6 Questionable

SCPI Commands :

```
STATus:QUEStionable[:EVENt]
STATus:QUEStionable:CONDition
STATus:QUEStionable:ENABle
STATus:QUEStionable:PTRansition
STATus:QUEStionable:NTRansition
```

class QuestionableCls

Questionable commands group definition. 10 total commands, 1 Subgroups, 5 group commands

get_condition() → int

```
# SCPI: STATus:QUEStionable:CONDition
value: int = driver.status.questionable.get_condition()
```

No command help available

return
register_value: No help available

get_enable() → int

```
# SCPI: STATus:QUEStionable:ENABle
value: int = driver.status.questionable.get_enable()
```

No command help available

return
register_value: No help available

get_event() → int

```
# SCPI: STATus:QUEStionable[:EVENt]
value: int = driver.status.questionable.get_event()
```

No command help available

return
register_value: No help available

get_ntransition() → int

```
# SCPI: STATus:QUEStionable:NTRansition
value: int = driver.status.questionable.get_ntransition()
```

No command help available

return
register_value: No help available

get_ptransition() → int

```
# SCPI: STATus:QUEStionable:PTRansition
value: int = driver.status.questionable.get_ptransition()
```

No command help available

return
register_value: No help available

set_enable(*register_value: int*) → None

```
# SCPI: STATus:QUEStionable:ENABle
driver.status.questionable.set_enable(register_value = 1)
```

No command help available

param register_value
No help available

set_ntransition(*register_value: int*) → None

```
# SCPI: STATus:QUEStionable:NTRansition
driver.status.questionable.set_ntransition(register_value = 1)
```

No command help available

param register_value
No help available

set_ptransition(*register_value: int*) → None

```
# SCPI: STATus:QUEStionable:PTRansition
driver.status.questionable.set_ptransition(register_value = 1)
```

No command help available

param register_value
No help available

Cloning the Group

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

Subgroups

6.30.6.1 Bit<BitNr>

RepCap Settings

```
# Range: Nr8 .. Nr12
rc = driver.status.questionable.bit.repcap_bitNr_get()
driver.status.questionable.bit.repcap_bitNr_set(repcap.BitNr.Nr8)
```

class BitCls

Bit commands group definition. 5 total commands, 5 Subgroups, 0 group commands Repeated Capability: BitNr, default value after init: BitNr.Nr8

Cloning the Group

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

Subgroups

6.30.6.1.1 Condition

SCPI Command :

```
STATus:QUEStionable:BIT<bitno>:CONDition
```

class ConditionCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:QUEStionable:BIT<bitno>:CONDition
value: bool = driver.status.questionable.bit.condition.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

6.30.6.1.2 Enable

SCPI Command :

```
STATus:QUEStionable:BIT<bitno>:ENABle
```

class EnableCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:QUEStionable:BIT<bitno>:ENABle
value: bool = driver.status.questionable.bit.enable.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

set(*register_bit: bool*, *bitNr=BitNr.Default*) → None

```
# SCPI: STATus:QUEStionable:BIT<bitno>:ENABle
driver.status.questionable.bit.enable.set(register_bit = False, bitNr = repcap.
↪BitNr.Default)
```

No command help available

param register_bit
No help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

6.30.6.1.3 Event

SCPI Command :

```
STATus:QUEStionable:BIT<bitno>[:EVENt]
```

class EventCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:QUEStionable:BIT<bitno>[:EVENt]
value: bool = driver.status.questionable.bit.event.get(bitNr = repcap.BitNr.
↪Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

6.30.6.1.4 Ntransition

SCPI Command :

```
STATus:QUEStionable:BIT<bitno>:NTRansition
```

class NtransitionCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:QUEStionable:BIT<bitno>:NTRansition
value: bool = driver.status.questionable.bit.ntransition.get(bitNr = repcap.
↪BitNr.Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

set(*register_bit: bool*, *bitNr=BitNr.Default*) → None

```
# SCPI: STATus:QUEStionable:BIT<bitno>:NTRansition
driver.status.questionable.bit.ntransition.set(register_bit = False, bitNr =␣
↪repcap.BitNr.Default)
```

No command help available

param register_bit
No help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

6.30.6.1.5 Ptransition

SCPI Command :

```
STATus:QUEStionable:BIT<bitno>:PTRansition
```

class PtransitionCls

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

get(*bitNr=BitNr.Default*) → bool

```
# SCPI: STATus:QUEStionable:BIT<bitno>:PTRansition
value: bool = driver.status.questionable.bit.ptransition.get(bitNr = repcap.
↪BitNr.Default)
```

No command help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

return
register_bit: No help available

set(*register_bit: bool*, *bitNr=BitNr.Default*) → None

```
# SCPI: STATus:QUEStionable:BIT<bitno>:PTRansition
driver.status.questionable.bit.ptransition.set(register_bit = False, bitNr =␣
↪repcap.BitNr.Default)
```

No command help available

param register_bit
No help available

param bitNr
optional repeated capability selector. Default value: Nr8 (settable in the interface 'Bit')

6.30.7 Queue

SCPI Command :

```
STATus:QUEue[:NEXT]
```

class QueueCls

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

class NextStruct

Structure for reading output parameters. Fields:

- Error_Code: int: No parameter help available
- Error_Description: str: No parameter help available

get_next() → NextStruct

```
# SCPI: STATus:QUEue[:NEXT]
value: NextStruct = driver.status.queue.get_next()
```

No command help available

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

6.31 System

SCPI Commands :

```
SYSTem:DID
SYSTem:KLOCk
SYSTem:PRESet
SYSTem:PRESet:ALL
SYSTem:PRESet:BASE
SYSTem:RESet
SYSTem:RESet:ALL
SYSTem:RESet:BASE
SYSTem:VERSion
```

class SystemCls

System commands group definition. 109 total commands, 22 Subgroups, 9 group commands

get_did() → str

```
# SCPI: SYSTem:DID
value: str = driver.system.get_did()
```

No command help available

return
device_id: No help available

get_klock() → bool

```
# SCPI: SYSTem:KLOCk
value: bool = driver.system.get_klock()
```

No command help available

return
klock: No help available

get_version() → float

```
# SCPI: SYSTem:VERSion
value: float = driver.system.get_version()
```

Queries the SCPI version number to which the instrument complies.

return
version: '1999.0' is the final SCPI version.

preset(*appl_name_and_li_number: str = None*) → None

```
# SCPI: SYSTem:PRESet
driver.system.preset(appl_name_and_li_number = 'abc')
```

No command help available

param appl_name_and_li_number
No help available

preset_all() → None

```
# SCPI: SYSTem:PRESet:ALL
driver.system.preset_all()
```

No command help available

preset_all_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:PRESet:ALL
driver.system.preset_all_with_opc()
```

No command help available

Same as preset_all, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

preset_base() → None

```
# SCPI: SYSTem:PRESet:BASE
driver.system.preset_base()
```

No command help available

preset_base_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:PRESet:BASE
driver.system.preset_base_with_opc()
```

No command help available

Same as preset_base, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

reset(*appl_name_and_li_number: str = None*) → None

```
# SCPI: SYSTem:RESet
driver.system.reset(appl_name_and_li_number = 'abc')
```

No command help available

param appl_name_and_li_number
No help available

reset_all() → None

```
# SCPI: SYSTem:RESet:ALL
driver.system.reset_all()
```

Resets the entire instrument, including base settings and all applications, even applications of other than the used remote channel.

reset_all_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:RESet:ALL
driver.system.reset_all_with_opc()
```

Resets the entire instrument, including base settings and all applications, even applications of other than the used remote channel.

Same as reset_all, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

reset_base() → None

```
# SCPI: SYSTem:RESet:BASE
driver.system.reset_base()
```

Resets the base settings.

reset_base_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:RESet:BASE
driver.system.reset_base_with_opc()
```

Resets the base settings.

Same as reset_base, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

set_klock(*klock: bool*) → None

```
# SCPI: SYSTem:KLOCk
driver.system.set_klock(klock = False)
```

No command help available

param klock
No help available

Cloning the Group

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

Subgroups

6.31.1 Attenuation

class AttenuationCls

Attenuation commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.31.1.1 CorrectionTable

class CorrectionTableCls

CorrectionTable 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.system.attenuation.correctionTable.clone()
```

Subgroups

6.31.1.1.1 All

class AllCls

All 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.system.attenuation.correctionTable.all.clone()
```

Subgroups

6.31.1.1.1.1 Globale

SCPI Command :

```
DELete:SYSTem:ATTenuation:CTABle:ALL:GLOBal
```

class GlobaleCls

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

delete() → None

```
# SCPI: DELete:SYSTem:ATTenuation:CTABle:ALL:GLOBal
driver.system.attenuation.correctionTable.all.globale.delete()
```

Deletes all global correction tables.

delete_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: DELete:SYSTem:ATTenuation:CTABle:ALL:GLOBal
driver.system.attenuation.correctionTable.all.globale.delete_with_opc()
```

Deletes all global correction tables.

Same as delete, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.1.1.1.2 Tenvironment

SCPI Command :

```
DELete:SYSTem:ATTenuation:CTABle:ALL[:TENVironment]
```

class TenvironmentCls

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

delete() → None

```
# SCPI: DELete:SYSTem:ATTenuation:CTABle:ALL[:TENVironment]
driver.system.attenuation.correctionTable.all.tenvironment.delete()
```

Deletes all channel-specific correction tables.

delete_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: DELete:SYSTem:ATTenuation:CTABle:ALL[:TENVironment]
driver.system.attenuation.correctionTable.all.tenvironment.delete_with_opc()
```

Deletes all channel-specific correction tables.

Same as delete, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.1.1.2 Globale

SCPI Command :

```
DELete:SYSTem:ATTenuation:CTABle:GLOBal
```

class GlobaleCls

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

delete(*name: str*) → None

```
# SCPI: DELete:SYSTem:ATTenuation:CTABle:GLOBal
driver.system.attenuation.correctionTable.globale.delete(name = 'abc')
```

Deletes a global correction table.

param name
The name of the correction table to be deleted.

6.31.1.1.3 Tenvironment

SCPI Command :

```
DELete:SYSTem:ATTenuation:CTABle[:TENVironment]
```

class TenvironmentCls

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

delete(*name: str*) → None

```
# SCPI: DELete:SYSTem:ATTenuation:CTABle[:TENVironment]
driver.system.attenuation.correctionTable.tenvironment.delete(name = 'abc')
```

Deletes a channel-specific correction table.

param name
The name of the correction table to be deleted.

6.31.2 Base

SCPI Command :

```
SYSTem:BASE:RELiability
```

class BaseCls

Base commands group definition. 31 total commands, 8 Subgroups, 1 group commands

get_reliability() → int

```
# SCPI: SYSTem:BASE:RELiability
value: int = driver.system.base.get_reliability()
```

Returns a reliability value indicating errors detected by the base software.

return
value: For reliability indicator values, see 'Checking the reliability indicator'.

Cloning the Group

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

Subgroups

6.31.2.1 Device

SCPI Commands :

```
SYSTem:BASE:DEVice:SUBinst
SYSTem:BASE:DEVice:COUNt
SYSTem:BASE:DEVice:RESet
SYSTem:BASE:DEVice:MSCont
SYSTem:BASE:DEVice:MSCCount
```

class DeviceCls

Device commands group definition. 8 total commands, 3 Subgroups, 5 group commands

class SubinstStruct

Structure for reading output parameters. Fields:

- Cur_Sub_Inst: int: Device number of the addressed channel, as indicated in a VISA resource string for HiSLIP and as returned by *DEV?. Mapping: device number 0 = channel 1 = assigned instrument 1
- Sub_Inst_Count: int: Total number of channels into which the instrument is split.

get_count() → int

```
# SCPI: SYSTem:BASE:DEVice:COUNt
value: int = driver.system.base.device.get_count()
```

Splits the instrument into channels or assigns all hardware resources to a single channel. Send this command to the channel with the lowest number (device 0 / channel 1 / assigned instrument 1) . To assign/distribute the available hardware resources to the channels, a reboot is performed automatically after you have changed the number of channels.

return
count: Number of channels The allowed subset of values depends on the number of connected RRHs.

get_msc_count() → int

```
# SCPI: SYSTem:BASE:DEVice:MSCCount
value: int = driver.system.base.device.get_msc_count()
```

Returns the maximum number of channels into which the instrument can be split.

return
max_sc_count: The value 0 indicates that no split is possible.

get_mscont() → int

```
# SCPI: SYSTem:BASE:DEVice:MSCont
value: int = driver.system.base.device.get_mscont()
```

No command help available

return
max_si_count: No help available

get_subinst() → SubinstStruct

```
# SCPI: SYSTem:BASE:DEVice:SUBinst
value: SubinstStruct = driver.system.base.device.get_subinst()
```

Queries the device number of the addressed channel and the total number of existing channels.

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

reset() → None

```
# SCPI: SYSTem:BASE:DEVice:RESet
driver.system.base.device.reset()
```

No command help available

reset_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:BASE:DEVice:RESet
driver.system.base.device.reset_with_opc()
```

No command help available

Same as reset, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

set_count(*count: int*) → None

```
# SCPI: SYSTem:BASE:DEVice:COUNt
driver.system.base.device.set_count(count = 1)
```

Splits the instrument into channels or assigns all hardware resources to a single channel. Send this command to the channel with the lowest number (device 0 / channel 1 / assigned instrument 1) . To assign/distribute the available hardware resources to the channels, a reboot is performed automatically after you have changed the number of channels.

param count
Number of channels The allowed subset of values depends on the number of connected RRHs.

Cloning the Group

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

Subgroups

6.31.2.1.1 License

SCPI Command :

```
SYSTem:BASE:DEVice:LICense
```

class LicenseCls

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

class LicenseStruct

Response structure. Fields:

- Sw_Option: List[str]: No parameter help available
- License_Count: List[int]: No parameter help available
- Instrument: List[int]: No parameter help available

get() → LicenseStruct

```
# SCPI: SYSTem:BASE:DEVice:LICense
value: LicenseStruct = driver.system.base.device.license.get()
```

No command help available

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

set(*sw_option: List[str] = None*, *license_count: List[int] = None*, *instrument: List[int] = None*) → None

```
# SCPI: SYSTem:BASE:DEVice:LICense
driver.system.base.device.license.set(sw_option = ['abc1', 'abc2', 'abc3'],␣
↪license_count = [1, 2, 3], instrument = [1, 2, 3])
```

No command help available

param sw_option
No help available

param license_count
No help available

param instrument
No help available

6.31.2.1.2 Setup

SCPI Command :

```
SYSTem:BASE:DEVice:SETup
```

class SetupCls

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

class SetupStruct

Response structure. Fields:

- Absolute_Item_Name: List[str]: No parameter help available
- Instrument: List[int]: No parameter help available

get() → SetupStruct

```
# SCPI: SYSTem:BASE:DEVice:SETup
value: SetupStruct = driver.system.base.device.setup.get()
```

No command help available

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

set(*absolute_item_name: List[str] = None*, *instrument: List[int] = None*) → None

```
# SCPI: SYSTem:BASE:DEVice:SETup
driver.system.base.device.setup.set(absolute_item_name = ['abc1', 'abc2', 'abc3
↪'], instrument = [1, 2, 3])
```

No command help available

param absolute_item_name
No help available

param instrument
No help available

6.31.2.1.3 Split

SCPI Command :

```
SYSTem:BASE:DEVice:SPLit
```

class SplitCls

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

class SplitStruct

Response structure. Fields:

- Count: int: Number of channels
- Direction: enums.DirectionHv: Direction of the split

get() → SplitStruct

```
# SCPI: SYSTem:BASE:DEVice:SPLit
value: SplitStruct = driver.system.base.device.split.get()
```

Splits the instrument into channels or assigns all hardware resources to a single channel. Send this command to the channel with the lowest number (device 0 / channel 1 / assigned instrument 1) . To assign/distribute the available hardware resources to the channels, a reboot is performed automatically after you have changed the number of channels.

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

set(*count: int*, *direction: DirectionHv*) → None

```
# SCPI: SYSTem:BASE:DEVice:SPLit
driver.system.base.device.split.set(count = 1, direction = enums.DirectionHv.
↪HORizontal)
```

Splits the instrument into channels or assigns all hardware resources to a single channel. Send this command to the channel with the lowest number (device 0 / channel 1 / assigned instrument 1) . To assign/distribute the available hardware resources to the channels, a reboot is performed automatically after you have changed the number of channels.

param count
Number of channels

param direction
Direction of the split

6.31.2.2 Display

SCPI Commands :

```
SYSTem:BASE:DISPlay:MWINdow
SYSTem:BASE:DISPlay:COLorset
SYSTem:BASE:DISPlay:FONTset
SYSTem:BASE:DISPlay:ROLLkeymode
SYSTem:BASE:DISPlay:LANGuage
```

class DisplayCls

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

get_color_set() → ColorSet

```
# SCPI: SYSTem:BASE:DISPlay:COLorset
value: enums.ColorSet = driver.system.base.display.get_color_set()
```

No command help available

return
color_set: No help available

get_font_set() → FontType

```
# SCPI: SYSTem:BASE:DISPlay:FONTset
value: enums.FontType = driver.system.base.display.get_font_set()
```

No command help available

return
fonset: No help available

get_language() → DisplayLanguage

```
# SCPI: SYSTem:BASE:DISPlay:LANGuage
value: enums.DisplayLanguage = driver.system.base.display.get_language()
```

No command help available

return
language: No help available

get_mwindow() → bool

```
# SCPI: SYSTem:BASE:DISPlay:MWINdow
value: bool = driver.system.base.display.get_mwindow()
```

No command help available

return
on_off: No help available

get_rollkey_mode() → RollkeyMode

```
# SCPI: SYSTem:BASE:DISPlay:ROLLkeymode
value: enums.RollkeyMode = driver.system.base.display.get_rollkey_mode()
```

No command help available

return
rollkey_mode: No help available

set_color_set(*color_set: ColorSet*) → None

```
# SCPI: SYSTem:BASE:DISPlay:COLorset
driver.system.base.display.set_color_set(color_set = enums.ColorSet.DEF)
```

No command help available

param color_set
No help available

set_font_set(*fonset: FontType*) → None

```
# SCPI: SYSTem:BASE:DISPlay:FONTset
driver.system.base.display.set_font_set(fonset = enums.FontType.DEF)
```

No command help available

param fonset
No help available

set_language(*language: DisplayLanguage*) → None

```
# SCPI: SYSTem:BASE:DISPlay:LANGuage
driver.system.base.display.set_language(language = enums.DisplayLanguage.AR)
```

No command help available

param language
No help available

set_mwindow(*on_off: bool*) → None

```
# SCPI: SYSTem:BASE:DISPlay:MWINdow
driver.system.base.display.set_mwindow(on_off = False)
```

No command help available

param on_off
No help available

set_rollkey_mode(*rollkey_mode: RollkeyMode*) → None

```
# SCPI: SYSTem:BASE:DISPlay:ROLLkeymode
driver.system.base.display.set_rollkey_mode(rollkey_mode = enums.RollkeyMode.
↪CURSors)
```

No command help available

param rollkey_mode
No help available

6.31.2.3 IpSet

class IpSetCls

IpSet 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.base.ipSet.clone()
```

Subgroups

6.31.2.3.1 SubMonitor

class SubMonitorCls

SubMonitor 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.base.ipSet.subMonitor.clone()
```

Subgroups

6.31.2.3.1.1 Refresh

SCPI Command :

```
SYSTem:BASE:IPSet:SMONitor:REFResh
```

class RefreshCls

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

set() → None

```
# SCPI: SYSTem:BASE:IPSet:SMONitor:REFResh
driver.system.base.ipSet.subMonitor.refresh.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:BASE:IPSet:SMONitor:REFResh
driver.system.base.ipSet.subMonitor.refresh.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.2.4 Option

class OptionCls

Option 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.system.base.option.clone()
```

Subgroups

6.31.2.4.1 Description

SCPI Command :

```
SYSTem:BASE:OPTion:DESCription
```

class DescriptionCls

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

get(*product_type: ProductType = None*, *validity: ValidityScope = None*, *scope: ValidityScopeB = None*, *instrument_no: float = None*) → str

```
# SCPI: SYSTem:BASE:OPTion:DESCription
value: str = driver.system.base.option.description.get(product_type = enums.
↪ProductType.ALL, validity = enums.ValidityScope.ALL, scope = enums.
↪ValidityScopeB.INSTrument, instrument_no = 1.0)
```

No command help available

param product_type
No help available

param validity
No help available

param scope
No help available

param instrument_no
No help available

return
option_list: No help available

6.31.2.4.2 ListPy

SCPI Command :

```
SYSTem:BASE:OPTion:LIST
```

class ListPyCls

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

get(*product_type: ProductType = None*, *validity: ValidityScope = None*, *scope: ValidityScopeB = None*, *instrument_no: float = None*) → str

```
# SCPI: SYSTem:BASE:OPTion:LIST
value: str = driver.system.base.option.listPy.get(product_type = enums.
↪ProductType.ALL, validity = enums.ValidityScope.ALL, scope = enums.
↪ValidityScopeB.INSTrument, instrument_no = 1.0)
```

Returns a list of installed software options (licenses) , hardware options, software packages and firmware applications. The list can be filtered using the described parameters. If filtering results in an empty list, a '0' is returned.

INTRO_CMD_HELP: The meaning of the filter <Validity> depends on the <OptionType> as follows:

- A software option is valid if there is an active license key for it. The value 'FUNCtional' is not relevant.
- A hardware option is functional if the corresponding hardware and all its components can be used (no defect detected) . The value 'VALid' is not relevant.

- A firmware application is functional if the required hardware, software and license keys are available and functional. The value 'VALid' is not relevant.
- For software packages, the filter has no effect.

param product_type
No help available

param validity
List only functional entries or only valid entries. By default or if ALL is selected, the list is not filtered according to the validity.

param scope
No help available

param instrument_no
No help available

return
option_list: No help available

6.31.2.4.3 Version

SCPI Command :

```
SYSTem:BASE:OPTion:VERSion
```

class VersionCls

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

get(*applicname: str = None*) → str

```
# SCPI: SYSTem:BASE:OPTion:VERSion
value: str = driver.system.base.option.version.get(applicname = 'abc')
```

Returns version information for installed software packages.
INTRO_CMD_HELP: You can either query a list of all installed packages and their versions or you can query the version of a single package specified via <Application>:

- <Application> specified: A string is returned, indicating the version of the <Application>. If the specified <Application> is unknown / not installed, '0' is returned.
- <Application> omitted: A string is returned, containing a list of all installed software packages and their version in the format '<PackageName1>,<Version1>;<PackageName2>,<Version2>;...'

param applicname
Software package for which the version is queried.

return
option_list: Single version or list of applications and versions

6.31.2.5 Password

SCPI Command :

```
SYSTem:BASE:PASSword:CDISable
```

class PasswordCls

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

set_cdisable(*user_mode: UserRole*) → None

```
# SCPI: SYSTem:BASE:PASSword:CDISable
driver.system.base.password.set_cdisable(user_mode = enums.UserRole.ADMin)
```

No command help available

param user_mode
No help available

Cloning the Group

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

Subgroups

6.31.2.5.1 Cenable

SCPI Commands :

```
SYSTem:BASE:PASSword[:CENable]:STATe
SYSTem:BASE:PASSword[:CENable]
```

class CenableCls

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

get_state() → UserRole

```
# SCPI: SYSTem:BASE:PASSword[:CENable]:STATe
value: enums.UserRole = driver.system.base.password.cenable.get_state()
```

No command help available

return
user_mode: No help available

set(*user_mode: UserRole*, *password: str*) → None

```
# SCPI: SYSTem:BASE:PASSword[:CENable]
driver.system.base.password.cenable.set(user_mode = enums.UserRole.ADMin,␣
↪password = 'abc')
```

No command help available

param user_mode
No help available

param password
No help available

6.31.2.6 Reference

class ReferenceCls

Reference 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.system.base.reference.clone()
```

Subgroups

6.31.2.6.1 Dc

class DcCls

Dc 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.base.reference.dc.clone()
```

Subgroups

6.31.2.6.1.1 Offset

SCPI Command :

```
SYSTem:BASE:REFerence:DC:OFFSet:ENABle
```

class OffsetCls

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

get_enable() → bool

```
# SCPI: SYSTem:BASE:REFerence:DC:OFFSet:ENABle
value: bool = driver.system.base.reference.dc.offset.get_enable()
```

No command help available

return
dc_offset_enable: No help available

set_enable(*dc_offset_enable: bool*) → None

```
# SCPI: SYSTem:BASE:REFerence:DC:OFFSet:ENABle
driver.system.base.reference.dc.offset.set_enable(dc_offset_enable = False)
```

No command help available

param dc_offset_enable
No help available

6.31.2.6.2 Frequency<Frequency>

RepCap Settings

```
# Range: Freq1 .. Freq4
rc = driver.system.base.reference.frequency.repcap_frequency_get()
driver.system.base.reference.frequency.repcap_frequency_set(repcap.Frequency.Freq1)
```

SCPI Commands :

```
SYSTem:BASE:REFerence:FREQuency:SOURce
SYSTem:BASE:REFerence:FREQuency
```

class FrequencyCls

Frequency commands group definition. 3 total commands, 1 Subgroups, 2 group commands Repeated Capability: Frequency, default value after init: Frequency.Freq1

get_source() → SourceIntExt

```
# SCPI: SYSTem:BASE:REFerence:FREQuency:SOURce
value: enums.SourceIntExt = driver.system.base.reference.frequency.get_source()
```

Selects the reference frequency source to be used.

return
source: No help available

get_value() → float

```
# SCPI: SYSTem:BASE:REFerence:FREQuency
value: float = driver.system.base.reference.frequency.get_value()
```

Queries the expected external reference frequency, for frequency source EXTernal.

return
ref_frequency: No help available

set_source(*source: SourceIntExt*) → None

```
# SCPI: SYSTem:BASE:REFerence:FREQuency:SOURce
driver.system.base.reference.frequency.set_source(source = enums.SourceIntExt.
↪EINTernal)
```

Selects the reference frequency source to be used.

param source
INTernal: Internal reference frequency EXTernal: External reference frequency

set_value(*ref_frequency: float*) → None

```
# SCPI: SYSTem:BASE:REFerence:FREQuency
driver.system.base.reference.frequency.set_value(ref_frequency = 1.0)
```

Queries the expected external reference frequency, for frequency source EXTernal.

param ref_frequency
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.base.reference.frequency.clone()
```

Subgroups

6.31.2.6.2.1 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.system.base.reference.frequency.advanced.clone()
```

Subgroups

6.31.2.6.2.2 Source

SCPI Command :

```
SYSTem:BASE:REFerence:FREQuency<n>:ADVanced:SOURce
```

class SourceCls

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

get(*frequency=Frequency.Default*) → SourceIntExt

```
# SCPI: SYSTem:BASE:REFerence:FREQuency<n>:ADVanced:SOURce
value: enums.SourceIntExt = driver.system.base.reference.frequency.advanced.
↪source.get(frequency = repcap.Frequency.Default)
```

No command help available

param frequency
optional repeated capability selector. Default value: Freq1 (settable in the interface 'Frequency')

return
source: No help available

set(*source: SourceIntExt*, *frequency=Frequency.Default*) → None

```
# SCPI: SYSTem:BASE:REFerence:FREQuency<n>:ADVanced:SOURce
driver.system.base.reference.frequency.advanced.source.set(source = enums.
↪SourceIntExt.EINTernal, frequency = repcap.Frequency.Default)
```

No command help available

param source
No help available

param frequency
optional repeated capability selector. Default value: Freq1 (settable in the interface 'Frequency')

6.31.2.6.3 Phase

SCPI Command :

```
SYSTem:BASE:REFerence:PHASe:OFFSet
```

class PhaseCls

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

get_offset() → float

```
# SCPI: SYSTem:BASE:REFerence:PHASe:OFFSet
value: float = driver.system.base.reference.phase.get_offset()
```

No command help available

return
phase_offset: No help available

set_offset(*phase_offset: float*) → None

```
# SCPI: SYSTem:BASE:REFerence:PHASe:OFFSet
driver.system.base.reference.phase.set_offset(phase_offset = 1.0)
```

No command help available

param phase_offset
No help available

6.31.2.7 Ssync

SCPI Commands :

```
SYSTem:BASE:SSYNc:MODE
SYSTem:BASE:SSYNc:OFFSet
```

class SsyncCls

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

get_mode() → CmwMode

```
# SCPI: SYSTem:BASE:SSYNc:MODE
value: enums.CmwMode = driver.system.base.ssync.get_mode()
```

No command help available

return
mode: No help available

get_offset() → int

```
# SCPI: SYSTem:BASE:SSYNc:OFFSet
value: int = driver.system.base.ssync.get_offset()
```

No command help available

return
offset: No help available

set_mode(*mode: CmwMode*) → None

```
# SCPI: SYSTem:BASE:SSYNc:MODE
driver.system.base.ssync.set_mode(mode = enums.CmwMode.GENerator)
```

No command help available

param mode
No help available

set_offset(*offset: int*) → None

```
# SCPI: SYSTem:BASE:SSYNc:OFFSet
driver.system.base.ssync.set_offset(offset = 1)
```

No command help available

param offset
No help available

6.31.2.8 StIcon

SCPI Commands :

```
SYSTem:BASE:STICon:ENABle
SYSTem:BASE:STICon:OPEN
SYSTem:BASE:STICon:CLOSe
```

class StIconCls

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

close() → None

```
# SCPI: SYSTem:BASE:STICon:CLOSe
driver.system.base.stIcon.close()
```

No command help available

close_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:BASE:STICon:CLOSe
driver.system.base.stIcon.close_with_opc()
```

No command help available

Same as close, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

get_enable() → bool

```
# SCPI: SYSTem:BASE:STICon:ENABle
value: bool = driver.system.base.stIcon.get_enable()
```

No command help available

return
on_off: No help available

open() → None

```
# SCPI: SYSTem:BASE:STICon:OPEN
driver.system.base.stIcon.open()
```

No command help available

open_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:BASE:STICon:OPEN
driver.system.base.stIcon.open_with_opc()
```

No command help available

Same as open, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

set_enable(*on_off: bool*) → None

```
# SCPI: SYSTem:BASE:STICon:ENABle
driver.system.base.stIcon.set_enable(on_off = False)
```

No command help available

param on_off
No help available

6.31.3 Cmw<CmwVariant>

RepCap Settings

```
# Range: Cmw1 .. Cmw100
rc = driver.system.cmw.repcap_cmwVariant_get()
driver.system.cmw.repcap_cmwVariant_set(repcap.CmwVariant.Cmw1)
```

class CmwCls

Cmw commands group definition. 3 total commands, 1 Subgroups, 0 group commands Repeated Capability: CmwVariant, default value after init: CmwVariant.Cmw1

Cloning the Group

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

Subgroups

6.31.3.1 Device

class DeviceCls

Device 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.system.cmw.device.clone()
```

Subgroups

6.31.3.1.1 Id

SCPI Command :

```
SYSTem:CMW<n>:DEVice:ID
```

class IdCls

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

get(*cmwVariant=CmwVariant.Default*) → str

```
# SCPI: SYSTem:CMW<n>:DEVice:ID
value: str = driver.system.cmw.device.id.get(cmwVariant = repcap.CmwVariant.
↪Default)
```

No command help available

param cmwVariant
optional repeated capability selector. Default value: Cmw1 (settable in the interface 'Cmw')

return
idn: No help available

6.31.3.1.2 Vi

SCPI Commands :

```
SYSTem:CMW:DEVice:VI:MODE
SYSTem:CMW:DEVice:VI:COUNt
```

class ViCls

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

get_count() → int

```
# SCPI: SYSTem:CMW:DEVice:VI:COUNt
value: int = driver.system.cmw.device.vi.get_count()
```

No command help available

return
vi_count: No help available

get_mode() → bool

```
# SCPI: SYSTem:CMW:DEVice:VI:MODE
value: bool = driver.system.cmw.device.vi.get_mode()
```

No command help available

return
vi_mode: No help available

6.31.4 Communicate

class CommunicateCls

Communicate commands group definition. 19 total commands, 7 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.31.4.1 Gpib<GpibInstance>

RepCap Settings

```
# Range: Inst1 .. Inst32
rc = driver.system.communicate.gpib.repcap_gpibInstance_get()
driver.system.communicate.gpib.repcap_gpibInstance_set(repcap.GpibInstance.Inst1)
```

class GpibCls

Gpib commands group definition. 3 total commands, 2 Subgroups, 0 group commands Repeated Capability: GpibInstance, default value after init: GpibInstance.Inst1

Cloning the Group

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

Subgroups

6.31.4.1.1 Self

class SelfCls

Self 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.system.communicate.gpib.self.clone()
```

Subgroups

6.31.4.1.1.1 Addr

SCPI Command :

```
SYSTem:COMMunicate:GPIB<inst>[:SELF]:ADDR
```

class AddrCls

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

get(*gpibInstance=GpibInstance.Default*) → int

```
# SCPI: SYSTem:COMMunicate:GPIB<inst>[:SELF]:ADDR
value: int = driver.system.communicate.gpib.self.addr.get(gpibInstance = repcap.
↪GpibInstance.Default)
```

No command help available

param gpibInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Gpib')

return
adress_no: No help available

set(*adress_no: int*, *gpibInstance=GpibInstance.Default*) → None

```
# SCPI: SYSTem:COMMunicate:GPIB<inst>[:SELF]:ADDR
driver.system.communicate.gpib.self.addr.set(adress_no = 1, gpibInstance =␣
↪repcap.GpibInstance.Default)
```

No command help available

param adress_no
No help available

param gpibInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Gpib')

6.31.4.1.1.2 Enable

SCPI Command :

```
SYSTem:COMMunicate:GPIB<inst>[:SELF]:ENABle
```

class EnableCls

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

get(*gpibInstance=GpibInstance.Default*) → bool

```
# SCPI: SYSTem:COMMunicate:GPIB<inst>[:SELF]:ENABle
value: bool = driver.system.communicate.gpib.self.enable.get(gpibInstance =␣
↪repcap.GpibInstance.Default)
```

No command help available

param gpibInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Gpib')

return
enable: No help available

set(*enable: bool*, *gpibInstance=GpibInstance.Default*) → None

```
# SCPI: SYSTem:COMMunicate:GPIB<inst>[:SELF]:ENABle
driver.system.communicate.gpib.self.enable.set(enable = False, gpibInstance =␣
↪repcap.GpibInstance.Default)
```

No command help available

param enable
No help available

param gpibInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Gpib')

6.31.4.1.2 Vresource

SCPI Command :

```
SYSTem:COMMunicate:GPIB<inst>:VRESource
```

class VresourceCls

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

get(*gpibInstance=GpibInstance.Default*) → str

```
# SCPI: SYSTem:COMMunicate:GPIB<inst>:VRESource
value: str = driver.system.communicate.gpib.vresource.get(gpibInstance = repcap.
↪GpibInstance.Default)
```

No command help available

param gpibInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Gpib')

return
visa_resource: No help available

6.31.4.2 Hislip<HislipInstance>

RepCap Settings

```
# Range: Inst1 .. Inst32
rc = driver.system.communicate.hislip.repcap_hislipInstance_get()
driver.system.communicate.hislip.repcap_hislipInstance_set(repcap.HislipInstance.Inst1)
```

class HislipCls

Hislip commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: HislipInstance, default value after init: HislipInstance.Inst1

Cloning the Group

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

Subgroups

6.31.4.2.1 Vresource

SCPI Command :

```
SYSTem:COMMunicate:HISLip<inst>:VRESource
```

class VresourceCls

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

get(*hislipInstance=HislipInstance.Default*) → str

```
# SCPI: SYSTem:COMMunicate:HISLip<inst>:VRESource
value: str = driver.system.communicate.hislip.vresource.get(hislipInstance =␣
↪repcap.HislipInstance.Default)
```

Queries the VISA resource string for the HiSLIP protocol.

param hislipInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Hislip')

return
visa_resource: No help available

6.31.4.3 Net

SCPI Commands :

```
SYSTem:COMMunicate:NET:ADAPter
SYSTem:COMMunicate:NET:GATeway
SYSTem:COMMunicate:NET:IPADdress
SYSTem:COMMunicate:NET:HOSTname
SYSTem:COMMunicate:NET:DHCP
```

class NetCls

Net commands group definition. 8 total commands, 2 Subgroups, 5 group commands

get_adapter() → str

```
# SCPI: SYSTem:COMMunicate:NET:ADAPter
value: str = driver.system.communicate.net.get_adapter()
```

Selects a LAN network adapter for configuration via other SYSTem:COMMunicate:NET... commands.

return
network_adapter: No help available

get_dhcp() → bool

```
# SCPI: SYSTem:COMMunicate:NET:DHCP
value: bool = driver.system.communicate.net.get_dhcp()
```

No command help available

return
dhcp_enable: No help available

get_gateway() → List[str]

```
# SCPI: SYSTem:COMMunicate:NET:GATeway
value: List[str] = driver.system.communicate.net.get_gateway()
```

Manually defines IPv4 addresses of default gateways. A query returns the currently defined addresses, irrespective of whether they have been specified manually or via DHCP.

return
gateways: No help available

get_hostname() → str

```
# SCPI: SYSTem:COMMunicate:NET:HOSTname
value: str = driver.system.communicate.net.get_hostname()
```

Queries the host name (computer name) of the CMX500. The host name is part of the VISA address string for LAN-based connections.

return
hostname: No help available

get_ip_address() → List[str]

```
# SCPI: SYSTem:COMMunicate:NET:IPADdress
value: List[str] = driver.system.communicate.net.get_ip_address()
```

Manually assigns one or more IPv4 addresses to the network adapter. A query returns the currently assigned addresses, irrespective of whether they have been assigned manually or via DHCP.

return
ip_addresses: No help available

set_adapter(*network_adapter: str*) → None

```
# SCPI: SYSTem:COMMunicate:NET:ADAPter
driver.system.communicate.net.set_adapter(network_adapter = 'abc')
```

Selects a LAN network adapter for configuration via other SYSTem:COMMunicate:NET… commands.

param network_adapter
No help available

set_dhcp(*dhcp_enable: bool*) → None

```
# SCPI: SYSTem:COMMunicate:NET:DHCP
driver.system.communicate.net.set_dhcp(dhcp_enable = False)
```

No command help available

param dhcp_enable
No help available

set_gateway(*gateways: List[str]*) → None

```
# SCPI: SYSTem:COMMunicate:NET:GATeway
driver.system.communicate.net.set_gateway(gateways = ['abc1', 'abc2', 'abc3'])
```

Manually defines IPv4 addresses of default gateways. A query returns the currently defined addresses, irrespective of whether they have been specified manually or via DHCP.

param gateways
Gateway IPv4 address consisting of four blocks separated by dots. Several strings separated by commas can be entered or several addresses separated by commas can be included in one string.

set_hostname(*hostname: str*) → None

```
# SCPI: SYSTem:COMMunicate:NET:HOSTname
driver.system.communicate.net.set_hostname(hostname = 'abc')
```

Queries the host name (computer name) of the CMX500. The host name is part of the VISA address string for LAN-based connections.

param hostname
No help available

set_ip_address(*ip_addresses: List[str]*) → None

```
# SCPI: SYSTem:COMMunicate:NET:IPADdress
driver.system.communicate.net.set_ip_address(ip_addresses = ['abc1', 'abc2',
↪'abc3'])
```

Manually assigns one or more IPv4 addresses to the network adapter. A query returns the currently assigned addresses, irrespective of whether they have been assigned manually or via DHCP.

param ip_addresses
The IPv4 address consisting of four blocks (octets) separated by dots. Several strings separated by commas can be entered or several addresses separated by commas can be included in one string.

Cloning the Group

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

Subgroups

6.31.4.3.1 Dns

SCPI Commands :

```
SYSTem:COMMunicate:NET:DNS:ENABle
SYSTem:COMMunicate:NET:DNS
```

class DnsCls

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

get_enable() → bool

```
# SCPI: SYSTem:COMMunicate:NET:DNS:ENABle
value: bool = driver.system.communicate.net.dns.get_enable()
```

No command help available

return
dns_enable: No help available

get_value() → List[str]

```
# SCPI: SYSTem:COMMunicate:NET:DNS
value: List[str] = driver.system.communicate.net.dns.get_value()
```

Manually defines the DNS server IPv4 addresses to be used. A query returns the defined DNS addresses, irrespective of whether they have been specified manually or via DHCP.

return
ip_addresses: No help available

set_enable(*dns_enable: bool*) → None

```
# SCPI: SYSTem:COMMunicate:NET:DNS:ENABle
driver.system.communicate.net.dns.set_enable(dns_enable = False)
```

No command help available

param dns_enable
No help available

set_value(*ip_addresses: List[str]*) → None

```
# SCPI: SYSTem:COMMunicate:NET:DNS
driver.system.communicate.net.dns.set_value(ip_addresses = ['abc1', 'abc2',
↪'abc3'])
```

Manually defines the DNS server IPv4 addresses to be used. A query returns the defined DNS addresses, irrespective of whether they have been specified manually or via DHCP.

param ip_addresses
DNS server IPv4 addresses consisting of four blocks separated by dots. Several strings separated by commas can be entered or several addresses separated by commas can be included in one string.

6.31.4.3.2 Subnet

SCPI Command :

```
SYSTem:COMMunicate:NET:SUBNet:MASK
```

class SubnetCls

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

get_mask() → List[str]

```
# SCPI: SYSTem:COMMunicate:NET:SUBNet:MASK
value: List[str] = driver.system.communicate.net.subnet.get_mask()
```

Manually defines the subnet masks to be used for the network adapter IPv4 addresses. A query returns the currently used subnet masks, irrespective of whether they have been assigned manually or via DHCP.

return
subnet_masks: No help available

set_mask(*subnet_masks: List[str]*) → None

```
# SCPI: SYSTem:COMMunicate:NET:SUBNet:MASK
driver.system.communicate.net.subnet.set_mask(subnet_masks = ['abc1', 'abc2',
↪'abc3'])
```

Manually defines the subnet masks to be used for the network adapter IPv4 addresses. A query returns the currently used subnet masks, irrespective of whether they have been assigned manually or via DHCP.

param subnet_masks
IPv4 subnet mask consisting of four blocks separated by dots. Several strings separated by commas can be entered or several masks separated by commas can be included in one string.

6.31.4.4 Rsib<RsibInstance>

RepCap Settings

```
# Range: Inst1 .. Inst32
rc = driver.system.communicate.rsib.repcap_rsibInstance_get()
driver.system.communicate.rsib.repcap_rsibInstance_set(repcap.RsibInstance.Inst1)
```

class RsibCls

Rsib commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: RsibInstance, default value after init: RsibInstance.Inst1

Cloning the Group

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

Subgroups

6.31.4.4.1 Vresource

SCPI Command :

```
SYSTem:COMMunicate:RSIB<inst>:VRESource
```

class VresourceCls

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

get(*rsibInstance=RsibInstance.Default*) → str

```
# SCPI: SYSTem:COMMunicate:RSIB<inst>:VRESource
value: str = driver.system.communicate.rsib.vresource.get(rsibInstance = repcap.
↪RsibInstance.Default)
```

No command help available

param rsibInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Rsib')

return
visa_resource: No help available

6.31.4.5 Socket<SocketInstance>

RepCap Settings

```
# Range: Inst1 .. Inst32
rc = driver.system.communicate.socket.repcap_socketInstance_get()
driver.system.communicate.socket.repcap_socketInstance_set(repcap.SocketInstance.Inst1)
```

class SocketCls

Socket commands group definition. 3 total commands, 3 Subgroups, 0 group commands Repeated Capability: SocketInstance, default value after init: SocketInstance.Inst1

Cloning the Group

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

Subgroups

6.31.4.5.1 Mode

SCPI Command :

```
SYSTem:COMMunicate:SOCKet<inst>:MODE
```

class ModeCls

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

get(*socketInstance=SocketInstance.Default*) → SocketProtocol

```
# SCPI: SYSTem:COMMunicate:SOCKet<inst>:MODE
value: enums.SocketProtocol = driver.system.communicate.socket.mode.
↪get(socketInstance = repcap.SocketInstance.Default)
```

Sets the protocol operation mode for direct socket communication.

param socketInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Socket')

return
protocol_mode: No help available

set(*protocol_mode: SocketProtocol*, *socketInstance=SocketInstance.Default*) → None

```
# SCPI: SYSTem:COMMunicate:SOCKet<inst>:MODE
driver.system.communicate.socket.mode.set(protocol_mode = enums.SocketProtocol.
↪AGILent, socketInstance = repcap.SocketInstance.Default)
```

Sets the protocol operation mode for direct socket communication.

param protocol_mode
RAW: no support of control messages AGILent: emulation codes via control connection (control port) IEEE1174: emulation codes via data connection (data port)

param socketInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Socket')

6.31.4.5.2 Port

SCPI Command :

```
SYSTem:COMMunicate:SOCKet<inst>:PORT
```

class PortCls

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

get(*socketInstance=SocketInstance.Default*) → int

```
# SCPI: SYSTem:COMMunicate:SOCKet<inst>:PORT
value: int = driver.system.communicate.socket.port.get(socketInstance = repcap.
↪SocketInstance.Default)
```

Sets the data port number for direct socket communication.

param socketInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Socket')

return
port_number: No help available

set(*port_number: int*, *socketInstance=SocketInstance.Default*) → None

```
# SCPI: SYSTem:COMMunicate:SOCKet<inst>:PORT
driver.system.communicate.socket.port.set(port_number = 1, socketInstance =␣
↪repcap.SocketInstance.Default)
```

Sets the data port number for direct socket communication.

param port_number
No help available

param socketInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Socket')

6.31.4.5.3 Vresource

SCPI Command :

```
SYSTem:COMMunicate:SOCKet<inst>:VRESource
```

class VresourceCls

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

get(*socketInstance=SocketInstance.Default*) → str

```
# SCPI: SYSTem:COMMunicate:SOCKet<inst>:VRESource
value: str = driver.system.communicate.socket.vresource.get(socketInstance =␣
↪repcap.SocketInstance.Default)
```

Queries the VISA resource string for direct socket communication.

param socketInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Socket')

return
visa_resource: No help available

6.31.4.6 Usb

SCPI Command :

```
SYSTem:COMMunicate:USB:VRESource
```

class UsbCls

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

get_vresource() → str

```
# SCPI: SYSTem:COMMunicate:USB:VRESource
value: str = driver.system.communicate.usb.get_vresource()
```

Queries the VISA resource string of the USB interface.

return
visa_resource: No help available

6.31.4.7 Vxi<VxiInstance>

RepCap Settings

```
# Range: Inst1 .. Inst32
rc = driver.system.communicate.vxi.repcap_vxiInstance_get()
driver.system.communicate.vxi.repcap_vxiInstance_set(repcap.VxiInstance.Inst1)
```

class VxiCls

Vxi commands group definition. 2 total commands, 2 Subgroups, 0 group commands Repeated Capability: VxiInstance, default value after init: VxiInstance.Inst1

Cloning the Group

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

Subgroups

6.31.4.7.1 Gtr

SCPI Command :

```
SYSTem:COMMunicate:VXI<inst>:GTR
```

class GtrCls

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

get(*vxiInstance=VxiInstance.Default*) → bool

```
# SCPI: SYSTem:COMMunicate:VXI<inst>:GTR
value: bool = driver.system.communicate.vxi.gtr.get(vxiInstance = repcap.
↪VxiInstance.Default)
```

Enables or disables the VXI-11 interface.

param vxiInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Vxi')

return
bool_switchremote: No help available

set(*bool_switchremote: bool*, *vxiInstance=VxiInstance.Default*) → None

```
# SCPI: SYSTem:COMMunicate:VXI<inst>:GTR
driver.system.communicate.vxi.gtr.set(bool_switchremote = False, vxiInstance =␣
↪repcap.VxiInstance.Default)
```

Enables or disables the VXI-11 interface.

param bool_switchremote
ON | 1: VXI-11 enabled OFF | 0: VXI-11 disabled

param vxiInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Vxi')

6.31.4.7.2 Vresource

SCPI Command :

```
SYSTem:COMMunicate:VXI<inst>:VRESource
```

class VresourceCls

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

get(*vxiInstance=VxiInstance.Default*) → str

```
# SCPI: SYSTem:COMMunicate:VXI<inst>:VRESource
value: str = driver.system.communicate.vxi.vresource.get(vxiInstance = repcap.
↪VxiInstance.Default)
```

Queries the VISA resource string for the VXI-11 protocol.

param vxiInstance
optional repeated capability selector. Default value: Inst1 (settable in the interface 'Vxi')

return
visa_resource: No help available

6.31.5 Connector

class ConnectorCls

Connector 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.connector.clone()
```

Subgroups

6.31.5.1 Translation

SCPI Command :

```
SYSTem:CONNector:TRANslation
```

class TranslationCls

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

class GetStruct

Response structure. Fields:

- Virtual_Connector: str: No parameter help available
- Absolute_Connector: str: No parameter help available

get(*connector: str*) → GetStruct

```
# SCPI: SYSTem:CONNector:TRANslation
value: GetStruct = driver.system.connector.translation.get(connector = rawAbc)
```

No command help available

param connector
No help available

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

6.31.6 Date

SCPI Command :

```
SYSTem:DATE
```

class DateCls

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

class DateStruct

Response structure. Fields:

- Year: int: No parameter help available
- Month: int: No parameter help available
- Day: int: No parameter help available

get() → DateStruct

```
# SCPI: SYSTem:DATE
value: DateStruct = driver.system.date.get()
```

No command help available

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

set(*year: int*, *month: int*, *day: int*) → None

```
# SCPI: SYSTem:DATE
driver.system.date.set(year = 1, month = 1, day = 1)
```

No command help available

param year
No help available

param month
No help available

param day
No help available

Cloning the Group

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

Subgroups

6.31.6.1 Local

SCPI Command :

```
SYSTem:DATE:LOCal
```

class LocalCls

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

class LocalStruct

Response structure. Fields:

- Year: int: No parameter help available
- Month: int: No parameter help available
- Day: int: No parameter help available

get() → LocalStruct

```
# SCPI: SYSTem:DATE:LOCal
value: LocalStruct = driver.system.date.local.get()
```

Sets the local date of the operating system calendar.

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

set(*year: int*, *month: int*, *day: int*) → None

```
# SCPI: SYSTem:DATE:LOCal
driver.system.date.local.set(year = 1, month = 1, day = 1)
```

Sets the local date of the operating system calendar.

param year
No help available

param month
No help available

param day
No help available

6.31.6.2 Utc

SCPI Command :

```
SYSTem:DATE:UTC
```

class UtcCls

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

class UtcStruct

Response structure. Fields:

- Year: int: No parameter help available
- Month: int: No parameter help available
- Day: int: No parameter help available

get() → UtcStruct

```
# SCPI: SYSTem:DATE:UTC
value: UtcStruct = driver.system.date.utc.get()
```

Sets the UTC date of the operating system calendar. Configuration is only possible for the time source MANual, see method RsCMPX_Base.System.Time.source.

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

set(*year: int*, *month: int*, *day: int*) → None

```
# SCPI: SYSTem:DATE:UTC
driver.system.date.utc.set(year = 1, month = 1, day = 1)
```

Sets the UTC date of the operating system calendar. Configuration is only possible for the time source MANual, see method RsCMPX_Base.System.Time.source.

param year
No help available

param month
No help available

param day
No help available

6.31.7 Device

SCPI Command :

```
SYSTem:DEVice:ID
```

class DeviceCls

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

get_id() → str

```
# SCPI: SYSTem:DEVice:ID
value: str = driver.system.device.get_id()
```

Queries the device identification of the instrument. This ID is important for ordering licenses.

return
device_id: No help available

6.31.8 DeviceFootprint

SCPI Command :

```
SYSTem:DFPRint
```

class DeviceFootprintCls

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

get() → bytes

```
# SCPI: SYSTem:DFPRint
value: bytes = driver.system.deviceFootprint.get()
```

Generates an XML file with footprint information about the instrument.

return
xml_device_footprint: Block data element containing the XML file contents.

set(*path: str = None*) → None

```
# SCPI: SYSTem:DFPRint
driver.system.deviceFootprint.set(path = 'abc')
```

Generates an XML file with footprint information about the instrument.

param path
No help available

Cloning the Group

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

Subgroups

6.31.8.1 History

SCPI Command :

```
SYSTem:DFPRint:HISTory:COUNt
```

class HistoryCls

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

get_count() → int

```
# SCPI: SYSTem:DFPRint:HISTory:COUNt
value: int = driver.system.deviceFootprint.history.get_count()
```

No command help available

return
count: No help available

Cloning the Group

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

Subgroups

6.31.8.1.1 Entry

SCPI Command :

```
SYSTem:DFPRint:HISTory:ENTRy
```

class EntryCls

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

get(*index: int*) → bytes

```
# SCPI: SYSTem:DFPRint:HISTory:ENTRy
value: bytes = driver.system.deviceFootprint.history.entry.get(index = 1)
```

No command help available

param index
No help available

return
xml_device_footprint: No help available

6.31.9 Display

SCPI Command :

```
SYSTem:DISPlay:UPDate
```

class DisplayCls

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

get_update() → bool

```
# SCPI: SYSTem:DISPlay:UPDate
value: bool = driver.system.display.get_update()
```

No command help available

return
display_update: No help available

set_update(*display_update: bool*) → None

```
# SCPI: SYSTem:DISPlay:UPDate
driver.system.display.set_update(display_update = False)
```

No command help available

param display_update
No help available

Cloning the Group

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

Subgroups

6.31.9.1 Monitor

SCPI Command :

```
SYSTem:DISPlay:MONitor
```

class MonitorCls

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

set_value(*enable: bool*) → None

```
# SCPI: SYSTem:DISPlay:MONitor
driver.system.display.monitor.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.system.display.monitor.clone()
```

Subgroups

6.31.9.1.1 Off

SCPI Command :

```
SYSTem:DISPlay:MONitor:OFF
```

class OffCls

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

set() → None

```
# SCPI: SYSTem:DISPlay:MONitor:OFF
driver.system.display.monitor.off.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:DISPlay:MONitor:OFF
driver.system.display.monitor.off.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.10 Error

SCPI Commands :

```
SYSTem:ERRor:ALL
SYSTem:ERRor:COUNt
```

class ErrorCls

Error commands group definition. 4 total commands, 1 Subgroups, 2 group commands

class AllStruct

Structure for reading output parameters. Fields:

- Error_Number: int: No parameter help available
- Error_Text: str: No parameter help available

get_all() → AllStruct

```
# SCPI: SYSTem:ERRor:ALL
value: AllStruct = driver.system.error.get_all()
```

Queries and deletes all entries in the error queue.

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

get_count() → int

```
# SCPI: SYSTem:ERRor:COUNt
value: int = driver.system.error.get_count()
```

Queries the number of entries in the error queue.

return
error_count: No help available

Cloning the Group

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

Subgroups

6.31.10.1 Code

SCPI Commands :

```
SYSTem:ERRor:CODE:ALL
SYSTem:ERRor:CODE[:NEXT]
```

class CodeCls

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

get_all() → int

```
# SCPI: SYSTem:ERRor:CODE:ALL
value: int = driver.system.error.code.get_all()
```

Queries and deletes all entries in the error queue.

return
error_code: Comma-separated list of error numbers. The error descriptions are not returned. Positive error numbers are instrument-specific. Negative error numbers are reserved by the SCPI standard.

get_next() → int

```
# SCPI: SYSTem:ERRor:CODE[:NEXT]
value: int = driver.system.error.code.get_next()
```

Queries and deletes the oldest entry in the error queue.

return
error: Only the error number is returned, not the error description. Positive error numbers are instrument-specific. Negative error numbers are reserved by the SCPI standard.

6.31.11 Generator

class GeneratorCls

Generator 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.generator.clone()
```

Subgroups

6.31.11.1 All

class AllCls

All 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.generator.all.clone()
```

Subgroups

6.31.11.1.1 Off

SCPI Command :

```
SYSTem:GENerator:ALL:OFF
```

class OffCls

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

set(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:GENerator:ALL:OFF
driver.system.generator.all.off.set()
```

Switch off all signaling applications, generators or measurements.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.12 Help

class HelpCls

Help 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.system.help.clone()
```

Subgroups

6.31.12.1 Headers

SCPI Command :

```
SYSTem:HELP:HEADers
```

class HeadersCls

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

get(*parser: str = None*) → bytes

```
# SCPI: SYSTem:HELP:HEADers
value: bytes = driver.system.help.headers.get(parser = 'abc')
```

No command help available

param parser
No help available

return
headers: No help available

6.31.12.2 Status

SCPI Commands :

```
SYSTem:HELP:STATus:BITS
SYSTem:HELP:STATus[:REGister]
```

class StatusCls

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

get_bits() → List[str]

```
# SCPI: SYSTem:HELP:STATus:BITS
value: List[str] = driver.system.help.status.get_bits()
```

No command help available

return
bits: No help available

get_register() → List[str]

```
# SCPI: SYSTem:HELP:STATus[:REGister]
value: List[str] = driver.system.help.status.get_register()
```

No command help available

return
register: No help available

6.31.12.3 Syntax

SCPI Commands :

```
SYSTem:HELP:SYNTax
SYSTem:HELP:SYNTax:ALL
```

class SyntaxCls

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

get(*header: str*) → bytes

```
# SCPI: SYSTem:HELP:SYNTax
value: bytes = driver.system.help.syntax.get(header = 'abc')
```

No command help available

param header
No help available

return
syntax: No help available

get_all() → bytes

```
# SCPI: SYSTem:HELP:SYNTax:ALL
value: bytes = driver.system.help.syntax.get_all()
```

No command help available

return
syntax: No help available

6.31.13 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.system.measurement.clone()
```

Subgroups

6.31.13.1 All

class AllCls

All 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.measurement.all.clone()
```

Subgroups

6.31.13.1.1 Off

SCPI Command :

```
SYSTem:MEASurement:ALL:OFF
```

class OffCls

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

set(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:MEASurement:ALL:OFF
driver.system.measurement.all.off.set()
```

Switch off all signaling applications, generators or measurements.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.14 Password

class PasswordCls

Password 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.password.clone()
```

Subgroups

6.31.14.1 New

SCPI Command :

```
SYSTem:PASSword:NEW
```

class NewCls

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

set(*current_password: str*, *new_password: str*) → None

```
# SCPI: SYSTem:PASSword:NEW
driver.system.password.new.set(current_password = 'abc', new_password = 'abc')
```

No command help available

param current_password
No help available

param new_password
No help available

6.31.15 Record

class RecordCls

Record 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.system.record.clone()
```

Subgroups

6.31.15.1 Macro

class MacroCls

Macro 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.system.record.macro.clone()
```

Subgroups

6.31.15.1.1 File

SCPI Commands :

```
SYSTem:RECord:MACRo:FILE:STARt
SYSTem:RECord:MACRo:FILE:STOP
```

class FileCls

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

start(*macro_id: str*) → None

```
# SCPI: SYSTem:RECord:MACRo:FILE:STARt
driver.system.record.macro.file.start(macro_id = 'abc')
```

Starts recording of submitted commands into a macro file. If the file exists, it is overwritten. If the file does not exist, it is created.

param macro_id
Path and filename of the destination file on the instrument

stop() → None

```
# SCPI: SYSTem:RECord:MACRo:FILE:STOP
driver.system.record.macro.file.stop()
```

Stops recording of commands into a macro file.

stop_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:RECord:MACRo:FILE:STOP
driver.system.record.macro.file.stop_with_opc()
```

Stops recording of commands into a macro file.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.16 Routing

class RoutingCls

Routing 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.routing.clone()
```

Subgroups

6.31.16.1 Possible

SCPI Command :

```
SYSTem:ROUTing:POSSible
```

class PossibleCls

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

get(*item: str = None*) → List[str]

```
# SCPI: SYSTem:ROUTing:POSSible
value: List[str] = driver.system.routing.possible.get(item = 'abc')
```

No command help available

param item
No help available

return
routing: No help available

6.31.17 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.system.signaling.clone()
```

Subgroups

6.31.17.1 All

class AllCls

All 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.signaling.all.clone()
```

Subgroups

6.31.17.1.1 Off

SCPI Command :

```
SYSTem:SIGNaling:ALL:OFF
```

class OffCls

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

set(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:SIGNaling:ALL:OFF
driver.system.signaling.all.off.set()
```

Switch off all signaling applications, generators or measurements.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.18 SingleCmw

class SingleCmwCls

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

Cloning the Group

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

Subgroups

6.31.18.1 Device

SCPI Command :

```
SYSTem:CMWS:DEVice:ID
```

class DeviceCls

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

get_id() → str

```
# SCPI: SYSTem:CMWS:DEVice:ID
value: str = driver.system.singleCmw.device.get_id()
```

No command help available

return
idn: No help available

6.31.19 Startup

class StartupCls

Startup 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.startup.clone()
```

Subgroups

6.31.19.1 Prepare

SCPI Command :

```
SYSTem:STARtup:PREPare:FDEFault
```

class PrepareCls

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

get_fdefault() → bool

```
# SCPI: SYSTem:STARtup:PREPare:FDEFault
value: bool = driver.system.startup.prepare.get_fdefault()
```

No command help available

return
on_off: No help available

set_fdefault(*on_off: bool*) → None

```
# SCPI: SYSTem:STARtup:PREPare:FDEFault
driver.system.startup.prepare.set_fdefault(on_off = False)
```

No command help available

param on_off
No help available

6.31.20 Time

SCPI Commands :

```
SYSTem:TIME
SYSTem:TIME:SOURce
SYSTem:TIME:NTP
```

class TimeCls

Time commands group definition. 12 total commands, 4 Subgroups, 3 group commands

class TimeStruct

Response structure. Fields:

- Hour: int: No parameter help available
- Min_Py: int: No parameter help available
- Sec: int: No parameter help available

get() → TimeStruct

```
# SCPI: SYSTem:TIME
value: TimeStruct = driver.system.time.get()
```

No command help available

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

get_ntp() → str

```
# SCPI: SYSTem:TIME:NTP
value: str = driver.system.time.get_ntp()
```

Configures the NTP server address for the time source NTP, see method RsCMPX_Base.System.Time.source.

return
time_server: No help available

get_source() → TimeSource

```
# SCPI: SYSTem:TIME:SOURce
value: enums.TimeSource = driver.system.time.get_source()
```

Selects the source for the date and time information.

return
time_source: - MANual: Manual configuration via SYSTem:DATE[:UTC] and SYSTem:TIME[:UTC]. - NTP: NTP server configured via SYSTem:TIME:NTP.

set(*hour: int*, *min_py: int*, *sec: int*) → None

```
# SCPI: SYSTem:TIME
driver.system.time.set(hour = 1, min_py = 1, sec = 1)
```

No command help available

param hour
No help available

param min_py
No help available

param sec
No help available

set_ntp(*time_server: str*) → None

```
# SCPI: SYSTem:TIME:NTP
driver.system.time.set_ntp(time_server = 'abc')
```

Configures the NTP server address for the time source NTP, see method RsCMPX_Base.System.Time.source.

param time_server
No help available

set_source(*time_source: TimeSource*) → None

```
# SCPI: SYSTem:TIME:SOURce
driver.system.time.set_source(time_source = enums.TimeSource.MANual)
```

Selects the source for the date and time information.

param time_source

- MANual: Manual configuration via SYSTem:DATE[:UTC] and SYSTem:TIME[:UTC].
- NTP: NTP server configured via SYSTem:TIME:NTP.

Cloning the Group

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

Subgroups

6.31.20.1 DaylightSavingTime

SCPI Command :

```
SYSTem:TIME:DSTime:MODE
```

class DaylightSavingTimeCls

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

get_mode() → bool

```
# SCPI: SYSTem:TIME:DSTime:MODE
value: bool = driver.system.time.daylightSavingTime.get_mode()
```

Configures whether the operating system automatically adjusts its clock for daylight saving time (DST) or not. The rules defining when exactly the clock must be adjusted by which offset depend on the configured time zone, see method RsCMPX_Base.System.Time.DaylightSavingTime.Rule.value. If the automatism is disabled, the local time is calculated as: Local time = UTC + time zone offset (no DST offset)

return
dst: No help available

set_mode(*dst: bool*) → None

```
# SCPI: SYSTem:TIME:DSTime:MODE
driver.system.time.daylightSavingTime.set_mode(dst = False)
```

Configures whether the operating system automatically adjusts its clock for daylight saving time (DST) or not. The rules defining when exactly the clock must be adjusted by which offset depend on the configured time zone, see method RsCMPX_Base.System.Time.DaylightSavingTime.Rule.value. If the automatism is disabled, the local time is calculated as: Local time = UTC + time zone offset (no DST offset)

param dst
ON | 1: automatism enabled OFF | 0: automatism disabled

Cloning the Group

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

Subgroups

6.31.20.1.1 Rule

SCPI Commands :

```
SYSTem:TIME:DSTime:RULE:CATalog
SYSTem:TIME:DSTime:RULE
```

class RuleCls

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

get_catalog() → str

```
# SCPI: SYSTem:TIME:DSTime:RULE:CATalog
value: str = driver.system.time.daylightSavingTime.rule.get_catalog()
```

Returns all time-zone values that can be set via method RsCMPX_Base.System.Time.DaylightSavingTime.Rule.value.

return
cat: Comma-separated list of all supported values, one string per value.

get_value() → str

```
# SCPI: SYSTem:TIME:DSTime:RULE
value: str = driver.system.time.daylightSavingTime.rule.get_value()
```

Sets the time zone in the date and time settings of the operating system. The used daylight saving time (DST) rules depend on the configured time zone. So this setting influences the automatic adjustment of the local time and date for DST. See also method RsCMPX_Base.System.Time.DaylightSavingTime.mode. Modifying the time zone modifies also the configured time zone offset, see method RsCMPX_Base.System.Tzone.set.

return
rule: No help available

set_value(*rule: str*) → None

```
# SCPI: SYSTem:TIME:DSTime:RULE
driver.system.time.daylightSavingTime.rule.set_value(rule = 'abc')
```

Sets the time zone in the date and time settings of the operating system. The used daylight saving time (DST) rules depend on the configured time zone. So this setting influences the automatic adjustment of the local time and date for DST. See also method RsCMPX_Base.System.Time.DaylightSavingTime.mode. Modifying the time zone modifies also the configured time zone offset, see method RsCMPX_Base.System.Tzone.set.

param rule
To query a list of all supported strings, use method RsCMPX_Base.System.Time.DaylightSavingTime.Rule.catalog.

6.31.20.2 HrTimer

SCPI Command :

```
SYSTem:TIME:HRTimer:RELative
```

class HrTimerCls

HrTimer commands group definition. 4 total commands, 1 Subgroups, 1 group commands

set_relative(*duration: int*) → None

```
# SCPI: SYSTem:TIME:HRTimer:RELative
driver.system.time.hrTimer.set_relative(duration = 1)
```

This command starts a timer. After the specified timeout, an OPC is generated. When the timer expires, 'Operation Complete' is indicated. This event can be evaluated by polling, via a *OPC? or via *WAI.

param duration
No help available

Cloning the Group

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

Subgroups

6.31.20.2.1 Absolute

SCPI Commands :

```
SYSTem:TIME:HRTimer:ABSolute:CLEar
SYSTem:TIME:HRTimer:ABSolute
```

class AbsoluteCls

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

clear() → None

```
# SCPI: SYSTem:TIME:HRTimer:ABSolute:CLEar
driver.system.time.hrTimer.absolute.clear()
```

No command help available

clear_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:TIME:HRTimer:ABSolute:CLEar
driver.system.time.hrTimer.absolute.clear_with_opc()
```

No command help available

Same as clear, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

set_value(*duration: float*) → None

```
# SCPI: SYSTem:TIME:HRTimer:ABSolute
driver.system.time.hrTimer.absolute.set_value(duration = 1.0)
```

This command starts a timer. The timeout is specified relative to an already set timestamp, see method RsCMPX_Base.System. Time.HrTimer.Absolute.Set.set. When the timer expires, 'Operation Complete' is indicated. This event can be evaluated by polling, via a *OPC? or via *WAI.

param duration
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.system.time.hrTimer.absolute.clone()
```

Subgroups

6.31.20.2.1.1 Set

SCPI Command :

```
SYSTem:TIME:HRTimer:ABSolute:SET
```

class SetCls

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

class GetStruct

Response structure. Fields:

- Year: int: No parameter help available
- Month: int: No parameter help available
- Day: int: No parameter help available
- Hour: int: No parameter help available
- Min_Py: int: No parameter help available
- Sec: float: No parameter help available
- Msec: int: No parameter help available

get() → GetStruct

```
# SCPI: SYSTem:TIME:HRTimer:ABSolute:SET
value: GetStruct = driver.system.time.hrTimer.absolute.set.get()
```

This command sets a timestamp with the current system time. A timer can be started with a timeout relative to this timestamp, see method RsCMPX_Base.System.Time.HrTimer.Absolute.value. An existing timestamp is overwritten.

return

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

set() → None

```
# SCPI: SYSTem:TIME:HRTimer:ABSolute:SET
driver.system.time.hrTimer.absolute.set.set()
```

This command sets a timestamp with the current system time. A timer can be started with a timeout relative to this timestamp, see method RsCMPX_Base.System.Time.HrTimer.Absolute.value. An existing timestamp is overwritten.

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: SYSTem:TIME:HRTimer:ABSolute:SET
driver.system.time.hrTimer.absolute.set.set_with_opc()
```

This command sets a timestamp with the current system time. A timer can be started with a timeout relative to this timestamp, see method RsCMPX_Base.System.Time.HrTimer.Absolute.value. An existing timestamp is overwritten.

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.31.20.3 Local

SCPI Command :

```
SYSTem:TIME:LOCal
```

class LocalCls

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

class LocalStruct

Response structure. Fields:

- Hour: int: No parameter help available
- Minute: int: No parameter help available
- Second: int: No parameter help available

get() → LocalStruct

```
# SCPI: SYSTem:TIME:LOCal
value: LocalStruct = driver.system.time.local.get()
```

Sets the local time of the operating system clock.

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

set(*hour: int*, *minute: int*, *second: int*) → None

```
# SCPI: SYSTem:TIME:LOCal
driver.system.time.local.set(hour = 1, minute = 1, second = 1)
```

Sets the local time of the operating system clock.

param hour
No help available

param minute
No help available

param second
No help available

6.31.20.4 Utc

SCPI Command :

```
SYSTem:TIME:UTC
```

class UtcCls

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

class UtcStruct

Response structure. Fields:

- Hour: int: No parameter help available
- Minute: int: No parameter help available
- Second: int: No parameter help available

get() → UtcStruct

```
# SCPI: SYSTem:TIME:UTC
value: UtcStruct = driver.system.time.utc.get()
```

Sets the UTC time of the operating system clock. Configuration is only possible for the time source MANUAL, see method RsCMPX_Base.System.Time.source.

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

set(*hour: int*, *minute: int*, *second: int*) → None

```
# SCPI: SYSTem:TIME:UTC
driver.system.time.utc.set(hour = 1, minute = 1, second = 1)
```

Sets the UTC time of the operating system clock. Configuration is only possible for the time source MANUAL, see method RsCMPX_Base.System.Time.source.

param hour
No help available

param minute
No help available

param second
No help available

6.31.21 Tzone

SCPI Command :

```
SYSTem:TZONe
```

class TzoneCls

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

class TzoneStruct

Response structure. Fields:

- Hour: int: No parameter help available
- Minute: int: No parameter help available

get() → TzoneStruct

```
# SCPI: SYSTem:TZONe
value: TzoneStruct = driver.system.tzone.get()
```

Specifies the offset of the local time to the universal time coordinated (UTC) due to the time zone. There can be an additional offset due to daylight saving time (DST) . Changing the time zone (offset) does not affect an eventual DST offset or the time zone configured via method RsCMPX_Base.System.Time.DaylightSavingTime.Rule.value. The local time is calculated as: local time = UTC + time zone offset + DST offset

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

set(*hour: int*, *minute: int*) → None

```
# SCPI: SYSTem:TZONe
driver.system.tzone.set(hour = 1, minute = 1)
```

Specifies the offset of the local time to the universal time coordinated (UTC) due to the time zone. There can be an additional offset due to daylight saving time (DST) . Changing the time zone (offset) does not affect an eventual DST offset or the time zone configured via method RsCMPX_Base.System.Time.DaylightSavingTime.Rule.value. The local time is calculated as: local time = UTC + time zone offset + DST offset

param hour
No help available

param minute
No help available

6.31.22 Update

SCPI Command :

```
SYSTem:UPDate:DGRoup
```

class UpdateCls

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

get_dgroup() → str

```
# SCPI: SYSTem:UPDate:DGRoup
value: str = driver.system.update.get_dgroup()
```

Sets the Device Group that the instrument belongs to. For remote installation, this setting must match the corresponding setting in the R&S software distributor.

return
devicegroup: No help available

set_dgroup(*devicegroup: str*) → None

```
# SCPI: SYSTem:UPDate:DGRoup
driver.system.update.set_dgroup(devicegroup = 'abc')
```

Sets the Device Group that the instrument belongs to. For remote installation, this setting must match the corresponding setting in the R&S software distributor.

param devicegroup
No help available

6.32 Tenvironment

class TenvironmentCls

Tenvironment 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.tenvironment.clone()
```

Subgroups

6.32.1 Spath

SCPI Command :

```
DELete:TENVironment:SPATh
```

class SpathCls

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

delete(*name_signal_path: str*) → None

```
# SCPI: DELete:TENVironment:SPATh
driver.tenvironment.spath.delete(name_signal_path = 'abc')
```

Deletes a connection.

param name_signal_path
The name of the connection to be deleted.

6.33 Trace

class TraceCls

Trace 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.trace.clone()
```

Subgroups

6.33.1 Remote

class RemoteCls

Remote 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.trace.remote.clone()
```

Subgroups

6.33.1.1 Mode

class ModeCls

Mode 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.trace.remote.mode.clone()
```

Subgroups

6.33.1.1.1 Display

SCPI Commands :

```
TRACe:REMote:MODE:DISPlay:CLEar
TRACe:REMote:MODE:DISPlay:ENABle
```

class DisplayCls

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

clear() → None

```
# SCPI: TRACe:REMote:MODE:DISPlay:CLEar
driver.trace.remote.mode.display.clear()
```

No command help available

clear_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: TRACe:REMote:MODE:DISPlay:CLEar
driver.trace.remote.mode.display.clear_with_opc()
```

No command help available

Same as clear, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

get_enable() → RemoteTraceEnable

```
# SCPI: TRACe:REMote:MODE:DISPlay:ENABle
value: enums.RemoteTraceEnable = driver.trace.remote.mode.display.get_enable()
```

No command help available

return
benable: No help available

set_enable(*benable: RemoteTraceEnable*) → None

```
# SCPI: TRACe:REMote:MODE:DISPlay:ENABle
driver.trace.remote.mode.display.set_enable(benable = enums.RemoteTraceEnable.
↪ANALysis)
```

No command help available

param benable
No help available

6.33.1.1.2 File<FileNr>

RepCap Settings

```
# Range: Nr1 .. Nr2
rc = driver.trace.remote.mode.file.repcap_fileNr_get()
driver.trace.remote.mode.file.repcap_fileNr_set(repcap.FileNr.Nr1)
```

class FileCls

File commands group definition. 11 total commands, 11 Subgroups, 0 group commands Repeated Capability: FileNr, default value after init: FileNr.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trace.remote.mode.file.clone()
```

Subgroups

6.33.1.1.2.1 Dexecution

class DexecutionCls

Dexecution 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.trace.remote.mode.file.dexecution.clone()
```

Subgroups

6.33.1.1.2.2 Duration

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:DEXecution:DURation
```

class DurationCls

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

get(*fileNr=FileNr.Default*) → bool

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:DEXecution:DURation
value: bool = driver.trace.remote.mode.file.dexecution.duration.get(fileNr =␣
↪repcap.FileNr.Default)
```

No command help available

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

return
benabled: No help available

set(*benabled: bool*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:DEXecution:DURation
driver.trace.remote.mode.file.dexecution.duration.set(benabled = False, fileNr␣
↪= repcap.FileNr.Default)
```

No command help available

param benabled
No help available

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

6.33.1.1.2.3 Enable

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:ENABle
```

class EnableCls

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

get(*fileNr=FileNr.Default*) → bool

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:ENABle
value: bool = driver.trace.remote.mode.file.enable.get(fileNr = repcap.FileNr.
↪Default)
```

No command help available

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

return
benable: No help available

set(*benable: bool*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:ENABle
driver.trace.remote.mode.file.enable.set(benable = False, fileNr = repcap.
↪FileNr.Default)
```

No command help available

param benable
No help available

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

6.33.1.1.2.4 FilterPy

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:FILTer
```

class FilterPyCls

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

class FilterPyStruct

Structure for setting input parameters. Fields:

- Binput: bool: No parameter help available
- Boutput: bool: No parameter help available
- Berror: bool: No parameter help available
- Btrigger: bool: No parameter help available
- Bdevice_Clear: bool: No parameter help available
- Bstatus_Register: bool: No parameter help available
- Bconnection: bool: No parameter help available
- Bremote_Local_Events: bool: No parameter help available

get(*fileNr=FileNr.Default*) → FilterPyStruct

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:FILTer
value: FilterPyStruct = driver.trace.remote.mode.file.filterPy.get(fileNr =␣
↪repcap.FileNr.Default)
```

No command help available

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

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

set(*structure: FilterPyStruct*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:FILTer
structure = driver.trace.remote.mode.file.filterPy.FilterPyStruct()
structure.Binput: bool = False
structure.Boutput: bool = False
structure.Berror: bool = False
structure.Btrigger: bool = False
structure.Bdevice_Clear: bool = False
structure.Bstatus_Register: bool = False
structure.Bconnection: bool = False
structure.Bremote_Local_Events: bool = False
driver.trace.remote.mode.file.filterPy.set(structure, fileNr = repcap.FileNr.
↪Default)
```

No command help available

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

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

6.33.1.1.2.5 FormatPy

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:FORMat
```

class FormatPyCls

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

get(*fileNr=FileNr.Default*) → RemoteTraceFileFormat

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:FORMat
value: enums.RemoteTraceFileFormat = driver.trace.remote.mode.file.formatPy.
↪get(fileNr = repcap.FileNr.Default)
```

No command help available

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

return
eformat: No help available

set(*eformat: RemoteTraceFileFormat*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:FORMat
driver.trace.remote.mode.file.formatPy.set(eformat = enums.
↪RemoteTraceFileFormat.ASCii, fileNr = repcap.FileNr.Default)
```

No command help available

param eformat
No help available

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

6.33.1.1.2.6 Functions

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:FUNCtions
```

class FunctionsCls

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

get(*fileNr=FileNr.Default*) → bool

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:FUNCtions
value: bool = driver.trace.remote.mode.file.functions.get(fileNr = repcap.
↪FileNr.Default)
```

No command help available

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

return
benable: No help available

set(*benable: bool*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:FUNCtions
driver.trace.remote.mode.file.functions.set(benable = False, fileNr = repcap.
↪FileNr.Default)
```

No command help available

param benable
No help available

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

6.33.1.1.2.7 Name

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:NAME
```

class NameCls

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

get(*fileNr=FileNr.Default*) → str

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:NAME
value: str = driver.trace.remote.mode.file.name.get(fileNr = repcap.FileNr.
↪Default)
```

No command help available

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

return
bs_file_path: No help available

set(*bs_file_path: str*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:NAME
driver.trace.remote.mode.file.name.set(bs_file_path = 'abc', fileNr = repcap.
↪FileNr.Default)
```

No command help available

param bs_file_path
No help available

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

6.33.1.1.2.8 Parser

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:PARSer
```

class ParserCls

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

get(*fileNr=FileNr.Default*) → bool

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:PARSer
value: bool = driver.trace.remote.mode.file.parser.get(fileNr = repcap.FileNr.
↪Default)
```

No command help available

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

return
benable: No help available

set(*benable: bool*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:PARSer
driver.trace.remote.mode.file.parser.set(benable = False, fileNr = repcap.
↪FileNr.Default)
```

No command help available

param benable
No help available

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

6.33.1.1.2.9 Rpc

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:RPC
```

class RpcCls

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

get(*fileNr=FileNr.Default*) → bool

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:RPC
value: bool = driver.trace.remote.mode.file.rpc.get(fileNr = repcap.FileNr.
↪Default)
```

No command help available

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

return
benable: No help available

set(*benable: bool*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:RPC
driver.trace.remote.mode.file.rpc.set(benable = False, fileNr = repcap.FileNr.
↪Default)
```

No command help available

param benable
No help available

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

6.33.1.1.2.10 Size

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:SIZE
```

class SizeCls

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

get(*fileNr=FileNr.Default*) → int

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:SIZE
value: int = driver.trace.remote.mode.file.size.get(fileNr = repcap.FileNr.
↪Default)
```

No command help available

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

return
ifile_size: No help available

set(*ifile_size: int*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:SIZE
driver.trace.remote.mode.file.size.set(ifile_size = 1, fileNr = repcap.FileNr.
↪Default)
```

No command help available

param ifile_size
No help available

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

6.33.1.1.2.11 StartMode

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:STARtmode
```

class StartModeCls

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

get(*fileNr=FileNr.Default*) → RemoteTraceStartMode

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:STARtmode
value: enums.RemoteTraceStartMode = driver.trace.remote.mode.file.startMode.
↪get(fileNr = repcap.FileNr.Default)
```

No command help available

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

return
estart_mode: No help available

set(*estart_mode: RemoteTraceStartMode*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:STARtmode
driver.trace.remote.mode.file.startMode.set(estart_mode = enums.
↪RemoteTraceStartMode.AUTO, fileNr = repcap.FileNr.Default)
```

No command help available

param estart_mode
No help available

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

6.33.1.1.2.12 StopMode

SCPI Command :

```
TRACe:REMote:MODE:FILE<instrument>:STOPmode
```

class StopModeCls

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

get(*fileNr=FileNr.Default*) → RemoteTraceStopMode

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:STOPmode
value: enums.RemoteTraceStopMode = driver.trace.remote.mode.file.stopMode.
↪get(fileNr = repcap.FileNr.Default)
```

No command help available

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

return
estop_mode: No help available

set(*estop_mode: RemoteTraceStopMode*, *fileNr=FileNr.Default*) → None

```
# SCPI: TRACe:REMote:MODE:FILE<instrument>:STOPmode
driver.trace.remote.mode.file.stopMode.set(estop_mode = enums.
↪RemoteTraceStopMode.AUTO, fileNr = repcap.FileNr.Default)
```

No command help available

param estop_mode
No help available

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

6.34 Trigger

class TriggerCls

Trigger commands group definition. 77 total commands, 15 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.34.1 Base

class BaseCls

Base commands group definition. 11 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.34.1.1 Eout<Eout>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.trigger.base.eout.repcap_eout_get()
driver.trigger.base.eout.repcap_eout_set(repcap.Eout.Nr1)
```

class EoutCls

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

Cloning the Group

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

Subgroups

6.34.1.1.1 Catalog

class CatalogCls

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

Cloning the Group

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

Subgroups

6.34.1.1.1.1 Source

SCPI Command :

```
TRIGger:BASE:EOUT<n>:CATalog:SOURce
```

class SourceCls

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

get(*eout=Eout.Default*) → List[str]

```
# SCPI: TRIGger:BASE:EOUT<n>:CATalog:SOURce
value: List[str] = driver.trigger.base.eout.catalog.source.get(eout = repcap.
↪Eout.Default)
```

No command help available

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

return
source_list: No help available

6.34.1.1.2 Source

SCPI Command :

```
TRIGger:BASE:EOUT<n>:SOURce
```

class SourceCls

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

get(*eout=Eout.Default*) → str

```
# SCPI: TRIGger:BASE:EOUT<n>:SOURce
value: str = driver.trigger.base.eout.source.get(eout = repcap.Eout.Default)
```

No command help available

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

return
source: No help available

set(*source: str*, *eout=Eout.Default*) → None

```
# SCPI: TRIGger:BASE:EOUT<n>:SOURce
driver.trigger.base.eout.source.set(source = 'abc', eout = repcap.Eout.Default)
```

No command help available

param source
No help available

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

6.34.1.2 ExtA

SCPI Commands :

```
TRIGger:BASE:EXTA:SOURce
TRIGger:BASE:EXTA:DIRection
TRIGger:BASE:EXTA:SLOPe
```

class ExtACls

ExtA commands group definition. 4 total commands, 1 Subgroups, 3 group commands

get_direction() → DirectionIo

```
# SCPI: TRIGger:BASE:EXTA:DIRection
value: enums.DirectionIo = driver.trigger.base.extA.get_direction()
```

Configures the trigger connectors as input or output connectors.

return
direction: IN: Input connector OUT: Output connector

get_slope() → SignalSlope

```
# SCPI: TRIGger:BASE:EXTA:SLOPe
value: enums.SignalSlope = driver.trigger.base.extA.get_slope()
```

Specifies whether the rising edge or the falling edge of the trigger pulse is generated at the trigger event. The setting applies to output trigger signals provided at the trigger connectors.

return
slope: REDGe: Rising edge FEDGe: Falling edge

get_source() → str

```
# SCPI: TRIGger:BASE:EXTA:SOURce
value: str = driver.trigger.base.extA.get_source()
```

Selects the output trigger signals to be routed to the trigger connectors. A list of all supported values can be retrieved using TRIGger:BASE:EXTA|EXTB:CATalog:SOURce?.

return
source: No help available

set_direction(*direction: DirectionIo*) → None

```
# SCPI: TRIGger:BASE:EXTA:DIRection
driver.trigger.base.extA.set_direction(direction = enums.DirectionIo.IN)
```

Configures the trigger connectors as input or output connectors.

param direction
IN: Input connector OUT: Output connector

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:BASE:EXTA:SLOPe
driver.trigger.base.extA.set_slope(slope = enums.SignalSlope.FEDGe)
```

Specifies whether the rising edge or the falling edge of the trigger pulse is generated at the trigger event. The setting applies to output trigger signals provided at the trigger connectors.

param slope
REDGe: Rising edge FEDGe: Falling edge

set_source(*source: str*) → None

```
# SCPI: TRIGger:BASE:EXTA:SOURce
driver.trigger.base.extA.set_source(source = 'abc')
```

Selects the output trigger signals to be routed to the trigger connectors. A list of all supported values can be retrieved using TRIGger:BASE:EXTA|EXTB:CATalog:SOURce?.

param source
No help available

Cloning the Group

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

Subgroups

6.34.1.2.1 Catalog

SCPI Command :

```
TRIGger:BASE:EXTA:CATalog:SOURce
```

class CatalogCls

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

get_source() → List[str]

```
# SCPI: TRIGger:BASE:EXTA:CATalog:SOURce
value: List[str] = driver.trigger.base.extA.catalog.get_source()
```

Lists all trigger source values that can be set using TRIGger:BASE:EXTA|EXTB:SOURce.

return
source_list: Comma-separated list of strings, one string per supported value

6.34.1.3 ExtB

SCPI Commands :

```
TRIGger:BASE:EXTB:DIRection
TRIGger:BASE:EXTB:SOURce
TRIGger:BASE:EXTB:SLOPe
```

class ExtBCls

ExtB commands group definition. 4 total commands, 1 Subgroups, 3 group commands

get_direction() → DirectionIo

```
# SCPI: TRIGger:BASE:EXTB:DIRection
value: enums.DirectionIo = driver.trigger.base.extB.get_direction()
```

Configures the trigger connectors as input or output connectors.

return
direction: IN: Input connector OUT: Output connector

get_slope() → SignalSlope

```
# SCPI: TRIGger:BASE:EXTB:SLOPe
value: enums.SignalSlope = driver.trigger.base.extB.get_slope()
```

Specifies whether the rising edge or the falling edge of the trigger pulse is generated at the trigger event. The setting applies to output trigger signals provided at the trigger connectors.

return
slope: REDGe: Rising edge FEDGe: Falling edge

get_source() → str

```
# SCPI: TRIGger:BASE:EXTB:SOURce
value: str = driver.trigger.base.extB.get_source()
```

Selects the output trigger signals to be routed to the trigger connectors. A list of all supported values can be retrieved using TRIGger:BASE:EXTA|EXTB:CATalog:SOURce?.

return
source: No help available

set_direction(*direction: DirectionIo*) → None

```
# SCPI: TRIGger:BASE:EXTB:DIRection
driver.trigger.base.extB.set_direction(direction = enums.DirectionIo.IN)
```

Configures the trigger connectors as input or output connectors.

param direction
IN: Input connector OUT: Output connector

set_slope(*slope: SignalSlope*) → None

```
# SCPI: TRIGger:BASE:EXTB:SLOPe
driver.trigger.base.extB.set_slope(slope = enums.SignalSlope.FEDGe)
```

Specifies whether the rising edge or the falling edge of the trigger pulse is generated at the trigger event. The setting applies to output trigger signals provided at the trigger connectors.

param slope
REDGe: Rising edge FEDGe: Falling edge

set_source(*source: str*) → None

```
# SCPI: TRIGger:BASE:EXTB:SOURce
driver.trigger.base.extB.set_source(source = 'abc')
```

Selects the output trigger signals to be routed to the trigger connectors. A list of all supported values can be retrieved using TRIGger:BASE:EXTA|EXTB:CATalog:SOURce?.

param source
No help available

Cloning the Group

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

Subgroups

6.34.1.3.1 Catalog

SCPI Command :

```
TRIGger:BASE:EXTB:CATalog:SOURce
```

class CatalogCls

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

get_source() → List[str]

```
# SCPI: TRIGger:BASE:EXTB:CATalog:SOURce
value: List[str] = driver.trigger.base.extB.catalog.get_source()
```

Lists all trigger source values that can be set using TRIGger:BASE:EXTA|EXTB:SOURce.

return
source_list: Comma-separated list of strings, one string per supported value

6.34.1.4 Uinitiated<Trigger>

RepCap Settings

```
# Range: Trg1 .. Trg4
rc = driver.trigger.base.uinitiated.repcap_trigger_get()
driver.trigger.base.uinitiated.repcap_trigger_set(repcap.Trigger.Trg1)
```

class UinitiatedCls

Uinitiated commands group definition. 1 total commands, 1 Subgroups, 0 group commands Repeated Capability: Trigger, default value after init: Trigger.Trg1

Cloning the Group

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

Subgroups

6.34.1.4.1 Execute

SCPI Command :

```
TRIGger:BASE:UINitiated<n>:EXECute
```

class ExecuteCls

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

set(*trigger=Trigger.Default*) → None

```
# SCPI: TRIGger:BASE:UINitiated<n>:EXECute
driver.trigger.base.uinitiated.execute.set(trigger = repcap.Trigger.Default)
```

Initiates the generation of a User Initiated Trigger signal.

param trigger
optional repeated capability selector. Default value: Trg1 (settable in the interface 'Uinitiated')

set_with_opc(*trigger=Trigger.Default*, *opc_timeout_ms: int = -1*) → None

6.34.2 Bluetooth

class BluetoothCls

Bluetooth commands group definition. 8 total commands, 1 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.34.2.1 Measurement

class MeasurementCls

Measurement commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

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

Subgroups

6.34.2.1.1 BhRate

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:BHRate:SOURce
```

class BhRateCls

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

get_source() → str

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:BHRate:SOURce
value: str = driver.trigger.bluetooth.measurement.bhRate.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:BHRate:SOURce
driver.trigger.bluetooth.measurement.bhRate.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

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

Subgroups

6.34.2.1.1.1 Catalog

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:BHRate:CATalog:SOURce
```

class CatalogCls

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

get_source() → List[str]

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:BHRate:CATalog:SOURce
value: List[str] = driver.trigger.bluetooth.measurement.bhRate.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.2.1.2 Hdr

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:HDR:SOURce
```

class HdrCls

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

get_source() → str

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:HDR:SOURce
value: str = driver.trigger.bluetooth.measurement.hdr.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:HDR:SOURce
driver.trigger.bluetooth.measurement.hdr.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.bluetooth.measurement.hdr.clone()
```

Subgroups

6.34.2.1.2.1 Catalog

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:HDR:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:HDR:CATalog:SOURce
value: List[str] = driver.trigger.bluetooth.measurement.hdr.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.2.1.3 Hdrp

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:HDRP:SOURce
```

class HdrpCls

Hdrp commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:HDRP:SOURce
value: str = driver.trigger.bluetooth.measurement.hdrp.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:HDRP:SOURce
driver.trigger.bluetooth.measurement.hdrp.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.bluetooth.measurement.hdrp.clone()
```

Subgroups

6.34.2.1.3.1 Catalog

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:HDRP:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:HDRP:CATalog:SOURce
value: List[str] = driver.trigger.bluetooth.measurement.hdrp.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.2.1.4 MultiEval

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.bluetooth.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.bluetooth.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.bluetooth.measurement.multiEval.clone()
```

Subgroups

6.34.2.1.4.1 Catalog

SCPI Command :

```
TRIGger:BLUetooth:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:BLUetooth:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.bluetooth.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.3 Cdma

class CdmaCls

Cdma 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.trigger.cdma.clone()
```

Subgroups

6.34.3.1 Measurement

class MeasurementCls

Measurement 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.trigger.cdma.measurement.clone()
```

Subgroups

6.34.3.1.1 MultiEval

SCPI Command :

```
TRIGger:CDMA:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:CDMA:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.cdma.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:CDMA:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.cdma.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.cdma.measurement.multiEval.clone()
```

Subgroups

6.34.3.1.1.1 Catalog

SCPI Command :

```
TRIGger:CDMA:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:CDMA:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.cdma.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.4 Gprf

class GprfCls

Gprf 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.trigger.gprf.clone()
```

Subgroups

6.34.4.1 Generator

class GeneratorCls

Generator commands group definition. 6 total commands, 2 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gprf.generator.clone()
```

Subgroups

6.34.4.1.1 Arb

SCPI Command :

```
TRIGger:GPRF:GENerator<Instance>[:ARB]:SOURce
```

class ArbCls

Arb commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:GPRF:GENerator<Instance>[:ARB]:SOURce
value: str = driver.trigger.gprf.generator.arb.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:GPRF:GENerator<Instance>[:ARB]:SOURce
driver.trigger.gprf.generator.arb.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gprf.generator.arb.clone()
```

Subgroups

6.34.4.1.1.1 Catalog

SCPI Command :

```
TRIGger:GPRF:GENerator<Instance>[:ARB]:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:GPRF:GENerator<Instance>[:ARB]:CATalog:SOURce
value: List[str] = driver.trigger.gprf.generator.arb.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.4.1.2 Sequencer

class SequencerCls

Sequencer 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.trigger.gprf.generator.sequencer.clone()
```

Subgroups

6.34.4.1.2.1 IsMeas

SCPI Commands :

```
TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISMeas:CATalog
TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISMeas:SOURce
```

class IsMeasCls

IsMeas commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_catalog() → List[str]

```
# SCPI: TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISMeas:CATalog
value: List[str] = driver.trigger.gprf.generator.sequencer.isMeas.get_catalog()
```

No command help available

return
trigger: No help available

get_source() → List[str]

```
# SCPI: TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISMeas:SOURce
value: List[str] = driver.trigger.gprf.generator.sequencer.isMeas.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: List[str]*) → None

```
# SCPI: TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISMeas:SOURce
driver.trigger.gprf.generator.sequencer.isMeas.set_source(trigger = ['abc1',
↪'abc2', 'abc3'])
```

No command help available

param trigger
No help available

6.34.4.1.2.2 IsTrigger

SCPI Commands :

```
TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISTRigger:CATalog
TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISTRigger:SOURce
```

class IsTriggerCls

IsTrigger commands group definition. 2 total commands, 0 Subgroups, 2 group commands

get_catalog() → List[str]

```
# SCPI: TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISTRigger:CATalog
value: List[str] = driver.trigger.gprf.generator.sequencer.isTrigger.get_
↪catalog()
```

No command help available

return
trigger: No help available

get_source() → str

```
# SCPI: TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISTRigger:SOURce
value: str = driver.trigger.gprf.generator.sequencer.isTrigger.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:GPRF:GENerator<Instance>:SEQuencer:ISTRigger:SOURce
driver.trigger.gprf.generator.sequencer.isTrigger.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

6.34.4.2 Measurement

class MeasurementCls

Measurement commands group definition. 8 total commands, 4 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gprf.measurement.clone()
```

Subgroups

6.34.4.2.1 FftSpecAn

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:FFTSanalyzer:SOURce
```

class FftSpecAnCls

FftSpecAn commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:FFTSanalyzer:SOURce
value: str = driver.trigger.gprf.measurement.fftSpecAn.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:FFTSanalyzer:SOURce
driver.trigger.gprf.measurement.fftSpecAn.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gprf.measurement.fftSpecAn.clone()
```

Subgroups

6.34.4.2.1.1 Catalog

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:FFTSanalyzer:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:FFTSanalyzer:CATalog:SOURce
value: List[str] = driver.trigger.gprf.measurement.fftSpecAn.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.4.2.2 IqRecorder

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:IQRecorder:SOURce
```

class IqRecorderCls

IqRecorder commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:IQRecorder:SOURce
value: str = driver.trigger.gprf.measurement.iqRecorder.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:IQRecorder:SOURce
driver.trigger.gprf.measurement.iqRecorder.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gprf.measurement.iqRecorder.clone()
```

Subgroups

6.34.4.2.2.1 Catalog

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:IQRecorder:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:IQRecorder:CATalog:SOURce
value: List[str] = driver.trigger.gprf.measurement.iqRecorder.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.4.2.3 IqVsSlot

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:IQVSlot:SOURce
```

class IqVsSlotCls

IqVsSlot commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:IQVSlot:SOURce
value: str = driver.trigger.gprf.measurement.iqVsSlot.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:IQVSlot:SOURce
driver.trigger.gprf.measurement.iqVsSlot.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gprf.measurement.iqVsSlot.clone()
```

Subgroups

6.34.4.2.3.1 Catalog

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:IQVSlot:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:IQVSlot:CATalog:SOURce
value: List[str] = driver.trigger.gprf.measurement.iqVsSlot.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.4.2.4 Power

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:POWer:SOURce
```

class PowerCls

Power commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:POWer:SOURce
value: str = driver.trigger.gprf.measurement.power.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:POWer:SOURce
driver.trigger.gprf.measurement.power.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gprf.measurement.power.clone()
```

Subgroups

6.34.4.2.4.1 Catalog

SCPI Command :

```
TRIGger:GPRF:MEASurement<Instance>:POWer:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:GPRF:MEASurement<Instance>:POWer:CATalog:SOURce
value: List[str] = driver.trigger.gprf.measurement.power.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.5 Gsm

class GsmCls

Gsm 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.trigger.gsm.clone()
```

Subgroups

6.34.5.1 Measurement

class MeasurementCls

Measurement 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.trigger.gsm.measurement.clone()
```

Subgroups

6.34.5.1.1 MultiEval

SCPI Command :

```
TRIGger:GSM:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.gsm.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.gsm.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.gsm.measurement.multiEval.clone()
```

Subgroups

6.34.5.1.1.1 Catalog

SCPI Command :

```
TRIGger:GSM:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:GSM:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.gsm.measurement.multiEval.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.6 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.trigger.lte.clone()
```

Subgroups

6.34.6.1 Measurement

class MeasurementCls

Measurement 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.trigger.lte.measurement.clone()
```

Subgroups

6.34.6.1.1 MultiEval

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.lte.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.lte.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.lte.measurement.multiEval.clone()
```

Subgroups

6.34.6.1.1.1 Catalog

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.lte.measurement.multiEval.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.6.1.2 Prach

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:PRACh:SOURce
```

class PrachCls

Prach commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SOURce
value: str = driver.trigger.lte.measurement.prach.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:SOURce
driver.trigger.lte.measurement.prach.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.lte.measurement.prach.clone()
```

Subgroups

6.34.6.1.2.1 Catalog

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:PRACh:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:PRACh:CATalog:SOURce
value: List[str] = driver.trigger.lte.measurement.prach.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.6.1.3 Srs

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:SRS:SOURce
```

class SrsCls

Srs commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SOURce
value: str = driver.trigger.lte.measurement.srs.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:SOURce
driver.trigger.lte.measurement.srs.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.lte.measurement.srs.clone()
```

Subgroups

6.34.6.1.3.1 Catalog

SCPI Command :

```
TRIGger:LTE:MEASurement<Instance>:SRS:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:LTE:MEASurement<Instance>:SRS:CATalog:SOURce
value: List[str] = driver.trigger.lte.measurement.srs.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.7 LteDl

class LteDlCls

LteDl 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.trigger.lteDl.clone()
```

Subgroups

6.34.7.1 Measurement

class MeasurementCls

Measurement 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.trigger.lteDl.measurement.clone()
```

Subgroups

6.34.7.1.1 MultiEval

SCPI Command :

```
TRIGger:LTEDl:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:LTEDl:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.lteDl.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:LTEDl:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.lteDl.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.lteDl.measurement.multiEval.clone()
```

Subgroups

6.34.7.1.1.1 Catalog

SCPI Command :

```
TRIGger:LTEDl:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:LTEDl:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.lteDl.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.8 Niot

class NiotCls

Niot commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.niot.clone()
```

Subgroups

6.34.8.1 Measurement

class MeasurementCls

Measurement 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.trigger.niot.measurement.clone()
```

Subgroups

6.34.8.1.1 MultiEval

SCPI Command :

```
TRIGger:NIOT:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NIOT:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.niot.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NIOT:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.niot.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.niot.measurement.multiEval.clone()
```

Subgroups

6.34.8.1.1.1 Catalog

SCPI Command :

```
TRIGger:NIOT:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NIOT:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.niot.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.8.1.2 Prach

SCPI Command :

```
TRIGger:NIOT:MEASurement<Instance>:PRACh:SOURce
```

class PrachCls

Prach commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NIOT:MEASurement<Instance>:PRACh:SOURce
value: str = driver.trigger.niot.measurement.prach.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NIOT:MEASurement<Instance>:PRACh:SOURce
driver.trigger.niot.measurement.prach.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.niot.measurement.prach.clone()
```

Subgroups

6.34.8.1.2.1 Catalog

SCPI Command :

```
TRIGger:NIOT:MEASurement<Instance>:PRACh:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NIOT:MEASurement<Instance>:PRACh:CATalog:SOURce
value: List[str] = driver.trigger.niot.measurement.prach.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.9 NrDl

class NrDlCls

NrDl 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.trigger.nrDl.clone()
```

Subgroups

6.34.9.1 Measurement

class MeasurementCls

Measurement 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.trigger.nrDl.measurement.clone()
```

Subgroups

6.34.9.1.1 MultiEval

SCPI Command :

```
TRIGger:NRDL:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NRDL:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.nrDl.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NRDL:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.nrDl.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrDl.measurement.multiEval.clone()
```

Subgroups

6.34.9.1.1.1 Catalog

SCPI Command :

```
TRIGger:NRDL:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NRDL:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.nrDl.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.10 NrMmw

class NrMmwCls

NrMmw commands group definition. 4 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrMmw.clone()
```

Subgroups

6.34.10.1 Measurement

class MeasurementCls

Measurement 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.trigger.nrMmw.measurement.clone()
```

Subgroups

6.34.10.1.1 MultiEval

SCPI Command :

```
TRIGger:NRMMw:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NRMMw:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.nrMmw.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NRMMw:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.nrMmw.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrMmw.measurement.multiEval.clone()
```

Subgroups

6.34.10.1.1.1 Catalog

SCPI Command :

```
TRIGger:NRMMw:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NRMMw:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.nrMmw.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.10.1.2 Prach

SCPI Command :

```
TRIGger:NRMMw:MEASurement<Instance>:PRACh:SOURce
```

class PrachCls

Prach commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NRMMw:MEASurement<Instance>:PRACh:SOURce
value: str = driver.trigger.nrMmw.measurement.prach.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NRMMw:MEASurement<Instance>:PRACh:SOURce
driver.trigger.nrMmw.measurement.prach.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrMmw.measurement.prach.clone()
```

Subgroups

6.34.10.1.2.1 Catalog

SCPI Command :

```
TRIGger:NRMMw:MEASurement<Instance>:PRACh:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NRMMw:MEASurement<Instance>:PRACh:CATalog:SOURce
value: List[str] = driver.trigger.nrMmw.measurement.prach.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.11 NrSub

class NrSubCls

NrSub 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.trigger.nrSub.clone()
```

Subgroups

6.34.11.1 Measurement

class MeasurementCls

Measurement 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.trigger.nrSub.measurement.clone()
```

Subgroups

6.34.11.1.1 MultiEval

SCPI Command :

```
TRIGger:NRSub:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.nrSub.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.nrSub.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrSub.measurement.multiEval.clone()
```

Subgroups

6.34.11.1.1.1 Catalog

SCPI Command :

```
TRIGger:NRSub:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.nrSub.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.11.1.2 Prach

SCPI Command :

```
TRIGger:NRSub:MEASurement<Instance>:PRACh:SOURce
```

class PrachCls

Prach commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:PRACh:SOURce
value: str = driver.trigger.nrSub.measurement.prach.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:PRACh:SOURce
driver.trigger.nrSub.measurement.prach.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrSub.measurement.prach.clone()
```

Subgroups

6.34.11.1.2.1 Catalog

SCPI Command :

```
TRIGger:NRSub:MEASurement<Instance>:PRACh:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:PRACh:CATalog:SOURce
value: List[str] = driver.trigger.nrSub.measurement.prach.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.11.1.3 Srs

SCPI Command :

```
TRIGger:NRSub:MEASurement<Instance>:SRS:SOURce
```

class SrsCls

Srs commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:SRS:SOURce
value: str = driver.trigger.nrSub.measurement.srs.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:SRS:SOURce
driver.trigger.nrSub.measurement.srs.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.nrSub.measurement.srs.clone()
```

Subgroups

6.34.11.1.3.1 Catalog

SCPI Command :

```
TRIGger:NRSub:MEASurement<Instance>:SRS:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:NRSub:MEASurement<Instance>:SRS:CATalog:SOURce
value: List[str] = driver.trigger.nrSub.measurement.srs.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.12 Uwb

class UwbCls

Uwb 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.trigger.uwb.clone()
```

Subgroups

6.34.12.1 Measurement

class MeasurementCls

Measurement 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.trigger.uwb.measurement.clone()
```

Subgroups

6.34.12.1.1 MultiEval

SCPI Command :

```
TRIGger:UWB:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:UWB:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.uwb.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:UWB:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.uwb.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.uwb.measurement.multiEval.clone()
```

Subgroups

6.34.12.1.1.1 Catalog

SCPI Command :

```
TRIGger:UWB:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:UWB:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.uwb.measurement.multiEval.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.13 Wcdma

class WcdmaCls

Wcdma commands group definition. 10 total commands, 1 Subgroups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wcdma.clone()
```

Subgroups

6.34.13.1 Measurement

class MeasurementCls

Measurement 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.trigger.wcdma.measurement.clone()
```

Subgroups

6.34.13.1.1 MultiEval

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.wcdma.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.wcdma.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wcdma.measurement.multiEval.clone()
```

Subgroups

6.34.13.1.1.1 Catalog

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.wcdma.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.13.1.2 OlpControl

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:OLPControl:SOURce
```

class OlpControlCls

OlpControl commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:OLPControl:SOURce
value: str = driver.trigger.wcdma.measurement.olpControl.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:OLPControl:SOURce
driver.trigger.wcdma.measurement.olpControl.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wcdma.measurement.olpControl.clone()
```

Subgroups

6.34.13.1.2.1 Catalog

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:OLPControl:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:OLPControl:CATalog:SOURce
value: List[str] = driver.trigger.wcdma.measurement.olpControl.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.13.1.3 OoSync

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:OOSYnc:SOURce
```

class OoSyncCls

OoSync commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:OOSYnc:SOURce
value: str = driver.trigger.wcdma.measurement.ooSync.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:OOSYnc:SOURce
driver.trigger.wcdma.measurement.ooSync.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wcdma.measurement.ooSync.clone()
```

Subgroups

6.34.13.1.3.1 Catalog

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:OOSYnc:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:OOSYnc:CATalog:SOURce
value: List[str] = driver.trigger.wcdma.measurement.ooSync.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.13.1.4 Prach

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:PRACh:SOURce
```

class PrachCls

Prach commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:PRACh:SOURce
value: str = driver.trigger.wcdma.measurement.prach.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:PRACh:SOURce
driver.trigger.wcdma.measurement.prach.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wcdma.measurement.prach.clone()
```

Subgroups

6.34.13.1.4.1 Catalog

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:PRACh:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:PRACh:CATalog:SOURce
value: List[str] = driver.trigger.wcdma.measurement.prach.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.13.1.5 Tpc

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:TPC:SOURce
```

class TpcCls

Tpc commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:TPC:SOURce
value: str = driver.trigger.wcdma.measurement.tpc.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:TPC:SOURce
driver.trigger.wcdma.measurement.tpc.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wcdma.measurement.tpc.clone()
```

Subgroups

6.34.13.1.5.1 Catalog

SCPI Command :

```
TRIGger:WCDMa:MEASurement<Instance>:TPC:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:WCDMa:MEASurement<Instance>:TPC:CATalog:SOURce
value: List[str] = driver.trigger.wcdma.measurement.tpc.catalog.get_source()
```

No command help available

return
trigger: No help available

6.34.14 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.trigger.wlan.clone()
```

Subgroups

6.34.14.1 Measurement

class MeasurementCls

Measurement 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.trigger.wlan.measurement.clone()
```

Subgroups

6.34.14.1.1 MultiEval

SCPI Command :

```
TRIGger:WLAN:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.wlan.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.wlan.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wlan.measurement.multiEval.clone()
```

Subgroups

6.34.14.1.1.1 Catalog

SCPI Command :

```
TRIGger:WLAN:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:WLAN:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.wlan.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.34.15 Wpan

class WpanCls

Wpan 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.trigger.wpan.clone()
```

Subgroups

6.34.15.1 Measurement

class MeasurementCls

Measurement 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.trigger.wpan.measurement.clone()
```

Subgroups

6.34.15.1.1 MultiEval

SCPI Command :

```
TRIGger:WPAN:MEASurement<Instance>:MEValuation:SOURce
```

class MultiEvalCls

MultiEval commands group definition. 2 total commands, 1 Subgroups, 1 group commands

get_source() → str

```
# SCPI: TRIGger:WPAN:MEASurement<Instance>:MEValuation:SOURce
value: str = driver.trigger.wpan.measurement.multiEval.get_source()
```

No command help available

return
trigger: No help available

set_source(*trigger: str*) → None

```
# SCPI: TRIGger:WPAN:MEASurement<Instance>:MEValuation:SOURce
driver.trigger.wpan.measurement.multiEval.set_source(trigger = 'abc')
```

No command help available

param trigger
No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.trigger.wpan.measurement.multiEval.clone()
```

Subgroups

6.34.15.1.1.1 Catalog

SCPI Command :

```
TRIGger:WPAN:MEASurement<Instance>:MEValuation:CATalog:SOURce
```

class CatalogCls

Catalog commands group definition. 1 total commands, 0 Subgroups, 1 group commands

get_source() → List[str]

```
# SCPI: TRIGger:WPAN:MEASurement<Instance>:MEValuation:CATalog:SOURce
value: List[str] = driver.trigger.wpan.measurement.multiEval.catalog.get_
↪source()
```

No command help available

return
trigger: No help available

6.35 TriggerInvoke

SCPI Command :

```
*TRG
```

class TriggerInvokeCls

TriggerInvoke commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set() → None

```
# SCPI: *TRG
driver.triggerInvoke.set()
```

No command help available

set_with_opc(*opc_timeout_ms: int = -1*) → None

```
# SCPI: *TRG
driver.triggerInvoke.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCMPX_Base.utilities.opc_timeout_set() to set the timeout value.

param opc_timeout_ms
Maximum time to wait in milliseconds, valid only for this call.

6.36 Unit

SCPI Commands :

```
UNIT:CONDuctance
UNIT:CHARge
UNIT:CAPacity
UNIT:ENERgy
```

(continues on next page)

(continued from previous page)

```
UNIT:FREQuency
UNIT:RESistor
UNIT:VOLTage
UNIT:ANGLe
UNIT:LENGth
UNIT:CURRent
UNIT:POWer
UNIT:TEMPerature
UNIT:TIME
```

class UnitCls

Unit commands group definition. 13 total commands, 0 Subgroups, 13 group commands

get_angle() → DefaultUnitAngle

```
# SCPI: UNIT:ANGLe
value: enums.DefaultUnitAngle = driver.unit.get_angle()
```

No command help available

return
default_unit_angle: No help available

get_capacity() → DefaultUnitCapacity

```
# SCPI: UNIT:CAPacity
value: enums.DefaultUnitCapacity = driver.unit.get_capacity()
```

No command help available

return
default_unit_capacity: No help available

get_charge() → DefaultUnitCharge

```
# SCPI: UNIT:CHARge
value: enums.DefaultUnitCharge = driver.unit.get_charge()
```

No command help available

return
default_unit_charge: No help available

get_conductance() → DefaultUnitConductance

```
# SCPI: UNIT:CONDuctance
value: enums.DefaultUnitConductance = driver.unit.get_conductance()
```

No command help available

return
default_unit_conductance: No help available

get_current() → DefaultUnitCurrent

```
# SCPI: UNIT:CURRent
value: enums.DefaultUnitCurrent = driver.unit.get_current()
```

No command help available

return
default_unit_current: No help available

get_energy() → DefaultUnitEnergy

```
# SCPI: UNIT:ENERgy
value: enums.DefaultUnitEnergy = driver.unit.get_energy()
```

No command help available

return
default_unit_energy: No help available

get_frequency() → DefaultUnitFrequency

```
# SCPI: UNIT:FREQuency
value: enums.DefaultUnitFrequency = driver.unit.get_frequency()
```

No command help available

return
default_unit_frequency: No help available

get_length() → DefaultUnitLenght

```
# SCPI: UNIT:LENGth
value: enums.DefaultUnitLenght = driver.unit.get_length()
```

No command help available

return
default_unit_lenght: No help available

get_power() → DefaultUnitPower

```
# SCPI: UNIT:POWer
value: enums.DefaultUnitPower = driver.unit.get_power()
```

No command help available

return
default_unit_power: No help available

get_resistor() → DefaultUnitResistor

```
# SCPI: UNIT:RESistor
value: enums.DefaultUnitResistor = driver.unit.get_resistor()
```

No command help available

return
default_unit_resistor: No help available

get_temperature() → DefaultUnitTemperature

```
# SCPI: UNIT:TEMPerature
value: enums.DefaultUnitTemperature = driver.unit.get_temperature()
```

No command help available

return
default_unit_temperature: No help available

get_time() → DefaultUnitTime

```
# SCPI: UNIT:TIME
value: enums.DefaultUnitTime = driver.unit.get_time()
```

No command help available

return
default_unit_time: No help available

get_voltage() → DefaultUnitVoltage

```
# SCPI: UNIT:VOLTage
value: enums.DefaultUnitVoltage = driver.unit.get_voltage()
```

No command help available

return
default_unit_voltage: No help available

set_angle(*default_unit_angle: DefaultUnitAngle*) → None

```
# SCPI: UNIT:ANGLe
driver.unit.set_angle(default_unit_angle = enums.DefaultUnitAngle.DEG)
```

No command help available

param default_unit_angle
No help available

set_capacity(*default_unit_capacity: DefaultUnitCapacity*) → None

```
# SCPI: UNIT:CAPacity
driver.unit.set_capacity(default_unit_capacity = enums.DefaultUnitCapacity.AF)
```

No command help available

param default_unit_capacity
No help available

set_charge(*default_unit_charge: DefaultUnitCharge*) → None

```
# SCPI: UNIT:CHARge
driver.unit.set_charge(default_unit_charge = enums.DefaultUnitCharge.AC)
```

No command help available

param default_unit_charge
No help available

set_conductance(*default_unit_conductance: DefaultUnitConductance*) → None

```
# SCPI: UNIT:CONDuctance
driver.unit.set_conductance(default_unit_conductance = enums.
↪DefaultUnitConductance.ASIE)
```

No command help available

param default_unit_conductance
No help available

set_current(*default_unit_current: DefaultUnitCurrent*) → None

```
# SCPI: UNIT:CURRent
driver.unit.set_current(default_unit_current = enums.DefaultUnitCurrent.A)
```

No command help available

param default_unit_current
No help available

set_energy(*default_unit_energy: DefaultUnitEnergy*) → None

```
# SCPI: UNIT:ENERgy
driver.unit.set_energy(default_unit_energy = enums.DefaultUnitEnergy.AJ)
```

No command help available

param default_unit_energy
No help available

set_frequency(*default_unit_frequency: DefaultUnitFrequency*) → None

```
# SCPI: UNIT:FREQuency
driver.unit.set_frequency(default_unit_frequency = enums.DefaultUnitFrequency.
↪AHZ)
```

No command help available

param default_unit_frequency
No help available

set_length(*default_unit_lenght: DefaultUnitLenght*) → None

```
# SCPI: UNIT:LENGth
driver.unit.set_length(default_unit_lenght = enums.DefaultUnitLenght.AM)
```

No command help available

param default_unit_lenght
No help available

set_power(*default_unit_power: DefaultUnitPower*) → None

```
# SCPI: UNIT:POWer
driver.unit.set_power(default_unit_power = enums.DefaultUnitPower.AW)
```

No command help available

param default_unit_power
No help available

set_resistor(*default_unit_resistor: DefaultUnitResistor*) → None

```
# SCPI: UNIT:RESistor
driver.unit.set_resistor(default_unit_resistor = enums.DefaultUnitResistor.AOHM)
```

No command help available

param default_unit_resistor
No help available

set_temperature(*default_unit_temperature: DefaultUnitTemperature*) → None

```
# SCPI: UNIT:TEMPerature
driver.unit.set_temperature(default_unit_temperature = enums.
↪DefaultUnitTemperature.C)
```

No command help available

param default_unit_temperature
No help available

set_time(*default_unit_time: DefaultUnitTime*) → None

```
# SCPI: UNIT:TIME
driver.unit.set_time(default_unit_time = enums.DefaultUnitTime.AS)
```

No command help available

param default_unit_time
No help available

set_voltage(*default_unit_voltage: DefaultUnitVoltage*) → None

```
# SCPI: UNIT:VOLTage
driver.unit.set_voltage(default_unit_voltage = enums.DefaultUnitVoltage.AV)
```

No command help available

param default_unit_voltage
No help available

6.37 Write

class WriteCls

Write 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.write.clone()
```

Subgroups

6.37.1 Eeprom

class EepromCls

Eeprom 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.write.eeprom.clone()
```

Subgroups

6.37.1.1 Data

SCPI Command :

```
WRITe:EEPRom:DATA
```

class DataCls

Data commands group definition. 1 total commands, 0 Subgroups, 1 group commands

set(*path: str*, *reserve: str*) → None

```
# SCPI: WRITe:EEPRom:DATA
driver.write.eeprom.data.set(path = 'abc', reserve = 'abc')
```

No command help available

param path
No help available

param reserve
No help available

CHAPTER

SEVEN

RSCMPX_BASE UTILITIES

class Utilities

Common utility class. Utility functions common for all types of drivers.

Access snippet: `utils = RsCMPX_Base.utilities`

property logger: *ScpiLogger*

Scpi Logger interface, see *here*

Access snippet: `logger = RsCMPX_Base.utilities.logger`

property driver_version: str

Returns the instrument driver version.

property idn_string: str

Returns instrument's identification string - the response on the SCPI command *IDN?

property manufacturer: str

Returns manufacturer of the instrument.

property full_instrument_model_name: str

Returns the current instrument's full name e.g. 'FSW26'.

property instrument_model_name: str

Returns the current instrument's family name e.g. 'FSW'.

property supported_models: List[str]

Returns a list of the instrument models supported by this instrument driver.

property instrument_firmware_version: str

Returns instrument's firmware version.

property instrument_serial_number: str

Returns instrument's serial_number.

query_opc(*timeout: int = 0*) → int

SCPI command: *OPC? Queries the instrument's OPC bit and hence it waits until the instrument reports operation complete. If you define timeout > 0, the VISA timeout is set to that value just for this method call.

property instrument_status_checking: bool

Sets / returns Instrument Status Checking. When True (default is True), all the driver methods and properties are sending "SYSTem:ERRor?" at the end to immediately react on error that might have occurred. We recommend to keep the state checking ON all the time. Switch it OFF only in rare cases when you require maximum speed. The default state after initializing the session is ON.

property encoding: str

Returns string<=>bytes encoding of the session.

property opc_query_after_write: bool

Sets / returns Instrument *OPC? query sending after each command write. When True, (default is False) the driver sends *OPC? every time a write command is performed. Use this if you want to make sure your sequence is performed command-after-command.

property bin_float_numbers_format: BinFloatFormat

Sets / returns format of float numbers when transferred as binary data.

property bin_int_numbers_format: BinIntFormat

Sets / returns format of integer numbers when transferred as binary data.

clear_status() → None

Clears instrument's status system, the session's I/O buffers and the instrument's error queue.

query_all_errors() → List[str]

Queries and clears all the errors from the instrument's error queue. The method returns list of strings as error messages. If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty. If you want to include the error codes, call the query_all_errors_with_codes()

query_all_errors_with_codes() → List[Tuple[int, str]]

Queries and clears all the errors from the instrument's error queue. The method returns list of tuples (code: int, message: str). If no error is detected, the return value is None. The process is: querying 'SYSTem:ERRor?' in a loop until the error queue is empty.

property instrument_options: List[str]

Returns all the instrument options. The options are sorted in the ascending order starting with K-options and continuing with B-options.

reset() → None

SCPI command: *RST Sends *RST command + calls the clear_status().

default_instrument_setup() → None

Custom steps performed at the init and at the reset().

self_test(*timeout: int = None*) → Tuple[int, str]

SCPI command: *TST? Performs instrument's self-test. Returns tuple (code:int, message: str). Code 0 means the self-test passed. You can define the custom timeout in milliseconds. If you do not define it, the default selftest timeout is used (usually 60 secs).

is_connection_active() → bool

Returns true, if the VISA connection is active and the communication with the instrument still works.

reconnect(*force_close: bool = False*) → bool

If the connection is not active, the method tries to reconnect to the device If the connection is active, and force_close is False, the method does nothing. If the connection is active, and force_close is True, the method closes, and opens the session again. Returns True, if the reconnection has been performed.

property resource_name: int

Returns the resource name used in the constructor

property opc_timeout: int

Sets / returns timeout in milliseconds for all the operations that use OPC synchronization.

property visa_timeout: int

Sets / returns visa IO timeout in milliseconds.

property data_chunk_size: int

Sets / returns the maximum size of one block transferred during write/read operations

property visa_manufacturer: int

Returns the manufacturer of the current VISA session.

process_all_commands() → None

SCPI command: *WAI Stops further commands processing until all commands sent before *WAI have been executed.

write_str(*cmd: str*) → None

Writes the command to the instrument.

write(*cmd: str*) → None

This method is an alias to the write_str(). Writes the command to the instrument as string.

write_int(*cmd: str*, *param: int*) → None

Writes the command to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2'

write_int_with_opc(*cmd: str*, *param: int*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the integer parameter: e.g.: cmd = 'SELECT:INPUT' param = '2', result command = 'SELECT:INPUT 2' If you do not provide timeout, the method uses current opc_timeout.

write_float(*cmd: str*, *param: float*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6'

write_float_with_opc(*cmd: str*, *param: float*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'CENTER:FREQ' param = '10E6', result command = 'CENTER:FREQ 10E6' If you do not provide timeout, the method uses current opc_timeout.

write_bool(*cmd: str*, *param: bool*) → None

Writes the command to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON'

write_bool_with_opc(*cmd: str*, *param: bool*, *timeout: int = None*) → None

Writes the command with OPC to the instrument followed by the boolean parameter: e.g.: cmd = 'OUTPUT' param = 'True', result command = 'OUTPUT ON' If you do not provide timeout, the method uses current opc_timeout.

query_str(*query: str*) → str

Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query(*query: str*) → str

This method is an alias to the query_str(). Sends the query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit.

query_bool(*query: str*) → bool

Sends the query to the instrument and returns the response as boolean.

query_int(*query: str*) → int

Sends the query to the instrument and returns the response as integer.

query_float(*query: str*) → float

Sends the query to the instrument and returns the response as float.

write_str_with_opc(*cmd: str*, *timeout: int = None*) → None

Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

write_with_opc(*cmd: str*, *timeout: int = None*) → None

This method is an alias to the write_str_with_opc(). Writes the opc-synced command to the instrument. If you do not provide timeout, the method uses current opc_timeout.

query_str_with_opc(*query: str*, *timeout: int = None*) → str

Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_with_opc(*query: str*, *timeout: int = None*) → str

This method is an alias to the query_str_with_opc(). Sends the opc-synced query to the instrument and returns the response as string. The response is trimmed of any trailing LF characters and has no length limit. If you do not provide timeout, the method uses current opc_timeout.

query_bool_with_opc(*query: str*, *timeout: int = None*) → bool

Sends the opc-synced query to the instrument and returns the response as boolean. If you do not provide timeout, the method uses current opc_timeout.

query_int_with_opc(*query: str*, *timeout: int = None*) → int

Sends the opc-synced query to the instrument and returns the response as integer. If you do not provide timeout, the method uses current opc_timeout.

query_float_with_opc(*query: str*, *timeout: int = None*) → float

Sends the opc-synced query to the instrument and returns the response as float. If you do not provide timeout, the method uses current opc_timeout.

write_bin_block(*cmd: str*, *payload: bytes*) → None

Writes all the payload as binary data block to the instrument. The binary data header is added at the beginning of the transmission automatically, do not include it in the payload!!!

query_bin_block(*query: str*) → bytes

Queries binary data block to bytes. Throws an exception if the returned data was not a binary data. Returns data:bytes

query_bin_block_with_opc(*query: str*, *timeout: int = None*) → bytes

Sends a OPC-synced query and returns binary data block to bytes. If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_float_list(*query: str*) → List[float]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_float_list_with_opc(*query: str*, *timeout: int = None*) → List[float]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_or_ascii_int_list(*query: str*) → List[int]

Queries a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32).

query_bin_or_ascii_int_list_with_opc(*query: str*, *timeout: int = None*) → List[int]

Sends a OPC-synced query and reads a list of floating-point numbers that can be returned in ASCII format or in binary format. - For ASCII format, the list numbers are decoded as comma-separated values. - For Binary Format, the numbers are decoded based on the property BinFloatFormat, usually float 32-bit (FORM REAL,32). If you do not provide timeout, the method uses current opc_timeout.

query_bin_block_to_file(*query: str*, *file_path: str*, *append: bool = False*) → None

Queries binary data block to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data. Example for transferring a file from Instrument -> PC: query = f'MMEM:DATA? '{INSTR_FILE_PATH}'". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

query_bin_block_to_file_with_opc(*query: str*, *file_path: str*, *append: bool = False*, *timeout: int = None*) → None

Sends a OPC-synced query and writes the returned data to the provided file. If append is False, any existing file content is discarded. If append is True, the new content is added to the end of the existing file, or if the file does not exit, it is created. Throws an exception if the returned data was not a binary data.

write_bin_block_from_file(*cmd: str*, *file_path: str*) → None

Writes data from the file as binary data block to the instrument using the provided command. Example for transferring a file from PC -> Instrument: cmd = f'MMEM:DATA '{INSTR_FILE_PATH}',". Alternatively, use the dedicated methods for this purpose:

- `send_file_from_pc_to_instrument()`
- `read_file_from_instrument_to_pc()`

send_file_from_pc_to_instrument(*source_pc_file: str*, *target_instr_file: str*) → None

SCPI Command: MMEM:DATA

Sends file from PC to the instrument

read_file_from_instrument_to_pc(*source_instr_file: str*, *target_pc_file: str*, *append_to_pc_file: bool = False*) → None

SCPI Command: MMEM:DATA?

Reads file from instrument to the PC.

Set the `append_to_pc_file` to True if you want to append the read content to the end of the existing PC file

get_last_sent_cmd() → str

Returns the last commands sent to the instrument. Only works in simulation mode

go_to_local() → None

Puts the instrument into local state.

go_to_remote() → None

Puts the instrument into remote state.

get_lock() → RLock

Returns the thread lock for the current session.

By default:

- If you create standard new RsCMPX_Base instance with new VISA session, the session gets a new thread lock. You can assign it to other RsCMPX_Base sessions in order to share one physical instrument with a multi-thread access.
- If you create new RsCMPX_Base from an existing session, the thread lock is shared automatically making both instances multi-thread safe.

You can always assign new thread lock by calling `driver.utilities.assign_lock()`

assign_lock(*lock: RLock*) → None

Assigns the provided thread lock.

clear_lock()

Clears the existing thread lock, making the current session thread-independent from others that might share the current thread lock.

sync_from(*source: Utilities*) → None

Synchronises these Utils with the source.

CHAPTER

EIGHT

RSCMPX_BASE LOGGER

Check the usage in the Getting Started chapter *here*.

class ScpiLogger

Base class for SCPI logging

mode

Sets the logging ON or OFF. Additionally, you can set the logging ON only for errors. Possible values:

- LoggingMode.Off - logging is switched OFF
- LoggingMode.On - logging is switched ON
- LoggingMode.Errors - logging is switched ON, but only for error entries
- LoggingMode.Default - sets the logging to default - the value you have set with logger.default_mode

default_mode

Sets / returns the default logging mode. You can recall the default mode by calling the logger.mode = LoggingMode.Default.

Data Type
LoggingMode

device_name: str

Use this property to change the resource name in the log from the default Resource Name (e.g. TCPIP::192.168.2.101::INSTR) to another name e.g. 'MySigGen1'.

set_logging_target(*target*, *console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target - the target must implement write() and flush(). You can optionally set the console and UDP logging ON or OFF. This method switches the logging target global OFF.

get_logging_target()

Based on the global_mode, it returns the logging target: either the local or the global one.

set_logging_target_global(*console_log: bool = None*, *udp_log: bool = None*) → None

Sets logging target to global. The global target must be defined. You can optionally set the console and UDP logging ON or OFF.

log_to_console

Returns logging to console status.

log_to_udp

Returns logging to UDP status.

log_to_console_and_udp

Returns true, if both logging to UDP and console in are True.

info_raw(*log_entry: str*, *add_new_line: bool = True*) → None

Method for logging the raw string without any formatting.

info(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one info entry. For binary log_string, use the info_bin()

error(*start_time: datetime*, *end_time: datetime*, *log_string_info: str*, *log_string: str*) → None

Method for logging one error entry.

set_relative_timestamp(*timestamp: datetime*) → None

If set, the further timestamps will be relative to the entered time.

set_relative_timestamp_now() → None

Sets the relative timestamp to the current time.

get_relative_timestamp() → datetime

Based on the global_mode, it returns the relative timestamp: either the local or the global one.

clear_relative_timestamp() → None

Clears the reference time, and the further logging continues with absolute times.

flush() → None

Flush all the entries.

log_status_check_ok

Sets / returns the current status of status checking OK. If True (default), the log contains logging of the status checking 'Status check: OK'. If False, the 'Status check: OK' is skipped - the log is more compact. Errors will still be logged.

clear_cached_entries() → None

Clears potential cached log entries. Cached log entries are generated when the Logging is ON, but no target has been defined yet.

set_format_string(*value: str*, *line_divider: str = '\n'*) → None

Sets new format string and line divider. If you just want to set the line divider, set the format string value=None The original format string is: `PAD_LEFT12(%START_TIME%) PAD_LEFT25(%DEVICE_NAME%) PAD_LEFT12(%DURATION%) %LOG_STRING_INFO%: %LOG_STRING%`

restore_format_string() → None

Restores the original format string and the line divider to LF

abbreviated_max_len_ascii: int

Defines the maximum length of one ASCII log entry. Default value is 200 characters.

abbreviated_max_len_bin: int

Defines the maximum length of one Binary log entry. Default value is 2048 bytes.

abbreviated_max_len_list: int

Defines the maximum length of one list entry. Default value is 100 elements.

bin_line_block_size: int

Defines number of bytes to display in one line. Default value is 16 bytes.

udp_port

Returns udp logging port.

target_auto_flushing

Returns status of the auto-flushing for the logging target.

CHAPTER

NINE

RSCMPX_BASE 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

A

B

C

D

E

F

G

H

I

L

M

P

R

S

T

U

W