RsCmwDau

Release 3.7.51.27

Rohde & Schwarz

May 27, 2021

CONTENTS:


CMW

CHAPTER

ONE

GETTING STARTED

1.1 Introduction

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

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

Check out this RsCmwBase example:

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

from RsCmwBase import *  # install from pypi.org

RsCmwBase.assert_minimum_version('3.7.90.32')
cmw_base = RsCmwBase('TCPIP::10.112.1.116::INSTR', True, False)
print(f'CMW Base IND: {cmw_base.utilities.idn_string}')
print(f'CMW Instrument options:\n{",".join(cmw_base.utilities.instrument_options)}')
cmw_base.utilities.visa_timeout = 5000

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

(continues on next page)

(continued from previous page)

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

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

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

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


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


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

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

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

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

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

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

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

1.2 Installation

RsCmwDau 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 RsCmwDau`

Option 2 - Installing in Pycharm

- In Pycharm Menu `File->Settings->Project->Project Interpreter` click on the **'+'** button on the bottom left
- Type `RsCmwDau` in the search box
- If you are behind a Proxy server, configure it in the Menu: `File->Settings->Appearance->System Settings->HTTP Proxy`

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

Option 3 - Offline Installation

If you are still reading the installation chapter, it is probably because the options above did not work for you - proxy problems, your boss saw the internet bill… Here are 5 easy step for installing the RsCmwDau offline:

- Download this python script (**Save target as**): rsinstrument_offline_install.py This installs all the preconditions that the RsCmwDau needs.
- Execute the script in your offline computer (supported is python 3.6 or newer)
- Download the RsCmwDau package to your computer from the pypi.org: https://pypi.org/project/RsCmwDau/#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\RsCmwDau-3.7.51.27.tar`

1.3 Finding Available Instruments

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

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

from RsCmwDau import *

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

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

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

1.4 Initiating Instrument Session

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

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

- Installed RsCmwDau Python module Version 3.7.51 or newer from pypi.org
- Installed VISA, for example R&S Visa 5.12 or newer
"""

from RsCmwDau import *

# A good practice is to assure that you have a certain minimum version installed
RsCmwDau.assert_minimum_version('3.7.51')
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 = RsCmwDau(resource_string_1)

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

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

Note: If you are wondering about the missing `ASRL1::INSTR`, yes, it works too, but come on… it's 2021.

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

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

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

from RsCmwDau import *

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

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

from RsCmwDau import *

driver = RsCmwDau('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 = RsCmwDau('TCPIP::192.168.56.101::HISLIP', True, True, "Simulate=True")
```

More `option_string` tokens are separated by comma:

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

Shared Session

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

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

from RsCmwDau import *

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

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

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

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

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

1.5 Plain SCPI Communication

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

driver = RsCmwDau('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 RsCmwDau import *

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

(continues on next page)

(continued from previous page)

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

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

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

After this time, the RsCmwDau raises an exception. Speaking of exceptions, an important feature of the RsCmwDau 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 RsCmwDau import *

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

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

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

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

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

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

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

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

1.6 Error Checking

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

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

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

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

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

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

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

See the next chapter on how to react on errors.

1.7 Exception Handling

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

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

(continues on next page)

(continued from previous page)

```
    driver = RsCmwDau('TCPIP::10.112.1.179::HISLIP')
except ResourceError as e:
    print(e.args[0])
    print('Your instrument is probably OFF...')
    # Exit now, no point of continuing
    exit(1)

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

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

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

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

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

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

Tip: General rules for exception handling:

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

1.8 Transferring Files

Instrument -> PC

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

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

1.9 Writing Binary Data

Writing from bytes

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

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

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

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

Writing from PC files

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

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

1.10 Transferring Big Data with Progress

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

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

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

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

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

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

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

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

1.11 Multithreading

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

import threading
from RsCmwDau import *


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


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

(continues on next page)

(continued from previous page)

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

driver.close()
```

Shared instrument session, accessed from multiple threads

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

Try the following example:

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

import threading
from RsCmwDau import *


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

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

# Wait for all threads to join this main thread
for t in threads:
    t.join()
```

(continues on next page)

(continued from previous page)

```
print('All threads ended')

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

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

Multiple instrument sessions accessed from multiple threads

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

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

Run the following example:

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

import threading
from RsCmwDau import *


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

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

threads = []
for i in range(10):
    t = threading.Thread(target=execute, args=(driver1, 1, i,))
    t.start()
    threads.append(t)
    t = threading.Thread(target=execute, args=(driver2, 2, i,))
```

(continues on next page)

(continued from previous page)

```
    t.start()
    threads.append(t)
print('All threads started')

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

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

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

CHAPTER

TWO

REVISION HISTORY

Rohde & Schwarz CMW Base System RsCmwBase instrument driver.

Supported instruments: CMW500, CMW100, CMW270, CMW280

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

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

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

Currently supported CMW subsystems:

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

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

---

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

Release Notes for the whole RsCmwXXX group:

Latest release notes summary: <INVALID>

Version 3.7.90.39

- <INVALID>

Version 3.8.xx2

- Fixed several misspelled arguments and command headers

Version 3.8.xx1

- Bluetooth and WLAN update for FW versions 3.8.xxx

Version 3.7.xx8

- Added documentation on ReadTheDocs

Version 3.7.xx7

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

Version 3.7.xx6

- Added new UDF integer number recognition

Version 3.7.xx5

- Added RsCmwDau

Version 3.7.xx4

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

Version 3.7.xx3

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

Version 3.7.xx2

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

Version 1.0.0.0

- First released version

CHAPTER

THREE

ENUMS

3.1 AddressModeA

```
# Example value:
value = enums.AddressModeA.AUTomatic
# All values (3x):
AUTomatic | DHCPv4 | STATic
```

3.2 AddressModeB

```
# Example value:
value = enums.AddressModeB.ACONf
# All values (3x):
ACONf | AUTO | STATic
```

3.3 AddressType

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

3.4 AkaVersion

```
# Example value:
value = enums.AkaVersion.AKA1
# All values (3x):
AKA1 | AKA2 | HTTP
```

3.5 AlignMode

```
# Example value:
value = enums.AlignMode.BANDwidtheff
# All values (2x):
BANDwidtheff | OCTetaligned
```

3.6 AmRnbBitrate

```
# First value:
value = enums.AmRnbBitrate.NOReq
# Last value:
value = enums.AmRnbBitrate.R795
# All values (9x):
NOReq | R1020 | R1220 | R475 | R515 | R590 | R670 | R740
R795
```

3.7 AmrType

```
# Example value:
value = enums.AmrType.NARRowband
# All values (2x):
NARRowband | WIDeband
```

3.8 AmRwbBitRate

```
# First value:
value = enums.AmRwbBitRate.NOReq
# Last value:
value = enums.AmRwbBitRate.RA2385
# All values (10x):
NOReq | R1265 | R1425 | R1585 | R1825 | R1985 | R2305 | R660
R885 | RA2385
```

3.9 ApplicationType

```
# First value:
value = enums.ApplicationType.AUDiodelay
# Last value:
value = enums.ApplicationType.THRoughput
# All values (9x):
AUDiodelay | DNSReq | IPANalysis | IPERf | IPLogging | IPReplay | OVERview | PING
THRoughput
```

3.10 AudioInstance

```
# Example value:
value = enums.AudioInstance.INST1
# All values (2x):
INST1 | INST2
```

3.11 AudioRouting

```
# Example value:
value = enums.AudioRouting.AUDioboard
# All values (3x):
AUDioboard | FORWard | LOOPback
```

3.12 AuthAlgorithm

```
# Example value:
value = enums.AuthAlgorithm.MILenage
# All values (2x):
MILenage | XOR
```

3.13 AuthScheme

```
# Example value:
value = enums.AuthScheme.AKA1
# All values (3x):
AKA1 | AKA2 | NOAuthentic
```

3.14 AvTypeA

```
# Example value:
value = enums.AvTypeA.AUDio
# All values (2x):
AUDio | VIDeo
```

3.15 AvTypeB

```
# Example value:
value = enums.AvTypeB.AUDio
# All values (3x):
AUDio | UNKNow | VIDeo
```

3.16 AvTypeC

```
# Example value:
value = enums.AvTypeC.AUDio
# All values (4x):
AUDio | EMER | UNK | VIDeo
```

3.17 Bandwidth

```
# Example value:
value = enums.Bandwidth.FB
# All values (7x):
FB | NB | NBFB | NBSWb | NBWB | SWB | WB
```

3.18 BehaviourA

```
# Example value:
value = enums.BehaviourA.AFTRng
# All values (7x):
AFTRng | ANSWer | BEFRng | BUSY | CD | DECLined | NOANswer
```

3.19 BehaviourB

```
# Example value:
value = enums.BehaviourB.FAILure
# All values (4x):
FAILure | NOACcept | NOANswer | NORMal
```

3.20 Bitrate

```
# First value:
value = enums.Bitrate.R1280
# Last value:
value = enums.Bitrate.R960
# All values (12x):
R1280 | R132 | R164 | R244 | R320 | R480 | R59 | R640
R72 | R80 | R96 | R960
```

3.21 BwRange

```
# Example value:
value = enums.BwRange.COMMon
# All values (2x):
COMMon | SENDrx
```

3.22 CallState

```
# Example value:
value = enums.CallState.CALLing
# All values (7x):
CALLing | CERRor | CESTablished | NOACtion | NOResponse | RELeased | RINGing
```

3.23 CallType

```
# Example value:
value = enums.CallType.ACK
# All values (8x):
ACK | GENeric | GPP | GPP2 | LARGe | PAGer | RCSChat | RCSGrpchat
```

3.24 ChawMode

```
# Example value:
value = enums.ChawMode.DIS
# All values (7x):
DIS | FIVE | NP | NUSed | SEVen | THRee | TWO
```

3.25 Cmr

```
# Example value:
value = enums.Cmr.DISable
# All values (4x):
DISable | ENABle | NP | PRESent
```

3.26 CodecType

```
# Example value:
value = enums.CodecType.EVS
# All values (3x):
EVS | NARRowband | WIDeband
```

3.27 ConnStatus

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

3.28 DataType

```
# Example value:
value = enums.DataType.AUDio
# All values (8x):
AUDio | CALL | FILetransfer | FTLMode | INValid | RCSLmsg | SMS | VIDeo
```

3.29 DauState

```
# Example value:
value = enums.DauState.ADJusted
# All values (7x):
ADJusted | AUTonomous | COUPled | INValid | OFF | ON | PENDing
```

3.30 DauStatus

```
# Example value:
value = enums.DauStatus.CONN
# All values (2x):
CONN | NOTConn
```

3.31 DirectionA

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

3.32 DirectionB

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

3.33 DtxRecv

```
# Example value:
value = enums.DtxRecv.DISable
# All values (3x):
DISable | ENABle | NP
```

3.34 EcallType

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

3.35 EvsBitrate

```
# First value:
value = enums.EvsBitrate.AW1265
# Last value:
value = enums.EvsBitrate.WLO7
# All values (41x):
AW1265 | AW1425 | AW1585 | AW1825 | AW1985 | AW2305 | AW66 | AW885
AWB2385 | NONE | NOReq | P1280 | P132 | P164 | P244 | P320
P480 | P640 | P960 | PR28 | PR59 | PR72 | PR80 | PR96
SDP | SHO2 | SHO3 | SHO5 | SHO7 | SLO2 | SLO3 | SLO5
SLO7 | WHO2 | WHO3 | WHO5 | WHO7 | WLO2 | WLO3 | WLO5
WLO7
```

3.36 EvsBw

```
# First value:
value = enums.EvsBw.DEAC
# Last value:
value = enums.EvsBw.WBCA
# All values (9x):
DEAC | FB | IO | NB | NOReq | SWB | SWBCa | WB
WBCA
```

3.37 EvsIoModeCnfg

```
# Example value:
value = enums.EvsIoModeCnfg.AMRWb
# All values (2x):
AMRWb | EVSamrwb
```

3.38 FileTransferType

```
# Example value:
value = enums.FileTransferType.FILetransfer
# All values (2x):
FILetransfer | LARGe
```

3.39 FilterConnect

```
# Example value:
value = enums.FilterConnect.BOTH
# All values (3x):
BOTH | CLOSed | OPEN
```

3.40 FilterType

```
# Example value:
value = enums.FilterType.APPL
# All values (8x):
APPL | CTRY | DSTP | FLOWid | IPADd | L4PR | L7PRotocol | SRCP
```

3.41 ForceModeEvs

```
# First value:
value = enums.ForceModeEvs.A1265
# Last value:
value = enums.ForceModeEvs.SDP
# All values (22x):
A1265 | A1425 | A1585 | A1825 | A1985 | A2305 | A2385 | A660
A885 | P1280 | P132 | P164 | P244 | P28 | P320 | P480
P640 | P72 | P80 | P96 | P960 | SDP
```

3.42 ForceModeNb

```
# First value:
value = enums.ForceModeNb.FIVE
# Last value:
value = enums.ForceModeNb.ZERO
# All values (9x):
FIVE | FOUR | FREE | ONE | SEVN | SIX | THRE | TWO
ZERO
```

3.43 ForceModeWb

```
# First value:
value = enums.ForceModeWb.EIGH
# Last value:
value = enums.ForceModeWb.ZERO
# All values (10x):
```

(continues on next page)

(continued from previous page)

```
EIGH | FIVE | FOUR | FREE | ONE | SEVN | SIX | THRE
TWO | ZERO
```

3.44 HfOnly

```
# Example value:
value = enums.HfOnly.BOTH
# All values (3x):
BOTH | HEADfull | NP
```

3.45 IdType

```
# Example value:
value = enums.IdType.ASND
# All values (7x):
ASND | ASNG | FQDN | IPVF | IPVS | KEY | RFC
```

3.46 InfoType

```
# Example value:
value = enums.InfoType.ERRor
# All values (4x):
ERRor | INFO | NONE | WARNing
```

3.47 IpSecEalgorithm

```
# Example value:
value = enums.IpSecEalgorithm.AES
# All values (4x):
AES | AUTO | DES | NOC
```

3.48 IpSecIalgorithm

```
# Example value:
value = enums.IpSecIalgorithm.AUTO
# All values (3x):
AUTO | HMMD | HMSH
```

3.49 IpV6AddLgh

```
# Example value:
value = enums.IpV6AddLgh.L16
# All values (2x):
L16 | L17
```

3.50 JitterDistrib

```
# Example value:
value = enums.JitterDistrib.NORMal
# All values (4x):
NORMal | PAReto | PNORmal | UNIForm
```

3.51 KeyType

```
# Example value:
value = enums.KeyType.OP
# All values (2x):
OP | OPC
```

3.52 Layer

```
# Example value:
value = enums.Layer.APP
# All values (5x):
APP | FEATure | L3 | L4 | L7
```

3.53 LoggingType

```
# Example value:
value = enums.LoggingType.LANDau
# All values (5x):
LANDau | UIPClient | UPIP | UPMulti | UPPP
```

3.54 MediaEndpoint

```
# Example value:
value = enums.MediaEndpoint.AUDioboard
# All values (4x):
AUDioboard | FORWard | LOOPback | PCAP
```

3.55 MobileStatus

```
# Example value:
value = enums.MobileStatus.EMERgency
# All values (5x):
EMERgency | EXPired | REGistered | TERMinated | UNRegistered
```

3.56 MtSmsEncoding

```
# Example value:
value = enums.MtSmsEncoding.BASE64
# All values (2x):
BASE64 | NENCoding
```

3.57 NetworkInterface

```
# Example value:
value = enums.NetworkInterface.IP
# All values (3x):
IP | LANDau | MULTicast
```

3.58 Origin

```
# Example value:
value = enums.Origin.MO
# All values (3x):
MO | MT | UNK
```

3.59 OverhUp

```
# Example value:
value = enums.OverhUp.FULL
# All values (3x):
FULL | NOK | OK
```

3.60 PauHeader

```
# Example value:
value = enums.PauHeader.COGE
# All values (8x):
COGE | CONF | CONRege | CORE | RECN | RECoge | REGD | REGE
```

3.61 PcapMode

```
# Example value:
value = enums.PcapMode.CYC
# All values (2x):
CYC | SING
```

3.62 PcScfStatus

```
# Example value:
value = enums.PcScfStatus.ERRor
# All values (6x):
ERRor | OFF | RUNNing | STARting | STOPping | UNKNown
```

3.63 PrefixType

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

3.64 Protocol

```
# Example value:
value = enums.Protocol.TCP
# All values (2x):
TCP | UDP
```

3.65 ProtocolB

```
# Example value:
value = enums.ProtocolB.ALL
# All values (3x):
ALL | TCP | UDP
```

3.66 QosMode

```
# Example value:
value = enums.QosMode.PRIO
# All values (2x):
PRIO | SAMeprio
```

3.67 ReceiveStatusA

```
# Example value:
value = enums.ReceiveStatusA.EMPT
# All values (3x):
EMPT | ERR | SUCC
```

3.68 ReceiveStatusB

```
# Example value:
value = enums.ReceiveStatusB.EMPT
# All values (3x):
EMPT | ERRO | SUCC
```

3.69 RegisterType

```
# Example value:
value = enums.RegisterType.IANA
# All values (2x):
IANA | OID
```

3.70 Repetition

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

3.71 ResourceState

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

3.72 Result

```
# Example value:
value = enums.Result.EMPT
# All values (4x):
EMPT | ERR | PEND | SUCC
```

3.73 RoutingType

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

3.74 ServerType

```
# Example value:
value = enums.ServerType.FOReign
# All values (4x):
FOReign | IAForeign | INTernal | NONE
```

3.75 ServiceTypeA

```
# Example value:
value = enums.ServiceTypeA.SERVer
# All values (2x):
SERVer | TGENerator
```

3.76 ServiceTypeB

```
# Example value:
value = enums.ServiceTypeB.BIDirectional
# All values (3x):
BIDirectional | CLIent | SERVer
```

3.77 SessionState

```
# First value:
value = enums.SessionState.BUSY
# Last value:
value = enums.SessionState.TERMinated
# All values (20x):
BUSY | CANCeled | CREated | DECLined | ESTablished | FILetransfer | HOLD | INITialmedia
MEDiaupdate | NOK | NONE | OK | PROGgres | RCSTxt | REJected | RELeased
RESumed | RINGing | SRVCcrelease | TERMinated
```

3.78 SessionUsage

```
# Example value:
value = enums.SessionUsage.OFF
# All values (3x):
OFF | ONALways | ONBYue
```

3.79 SignalingType

```
# Example value:
value = enums.SignalingType.EARLymedia
# All values (7x):
EARLymedia | NOPRecondit | PRECondit | REQU100 | REQuprecondi | SIMPle | WOTPrec183
```

3.80 SipTimerSel

```
# Example value:
value = enums.SipTimerSel.CUSTom
# All values (3x):
CUSTom | DEFault | RFC
```

3.81 SmsEncoding

```
# Example value:
value = enums.SmsEncoding.ASCI
# All values (7x):
ASCI | BASE64 | GSM7 | GSM8 | IAF5 | NENC | UCS
```

3.82 SmsStatus

```
# Example value:
value = enums.SmsStatus.NONE
# All values (4x):
NONE | SCOMpleted | SFAiled | SIPRogress
```

3.83 SmsTypeA

```
# Example value:
value = enums.SmsTypeA.OGPP
# All values (6x):
OGPP | OGPP2 | OPAGer | TGPP | TGPP2 | TPAGer
```

3.84 SmsTypeB

```
# Example value:
value = enums.SmsTypeB.TGP2
# All values (2x):
TGP2 | TGPP
```

3.85 SourceInt

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

3.86 StartMode

```
# Example value:
value = enums.StartMode.EAMRwbio
# All values (2x):
EAMRwbio | EPRimary
```

3.87 Testcall

```
# Example value:
value = enums.Testcall.FALSe
# All values (2x):
FALSe | TRUE
```

3.88 TestResult

```
# Example value:
value = enums.TestResult.FAILed
# All values (3x):
FAILed | NONE | SUCCeded
```

3.89 ThroughputType

```
# Example value:
value = enums.ThroughputType.OVERall
# All values (2x):
OVERall | RAN
```

3.90 TransportSel

```
# Example value:
value = enums.TransportSel.CUSTom
# All values (4x):
CUSTom | DEFault | TCP | UDP
```

3.91 UpdateCallEvent

```
# Example value:
value = enums.UpdateCallEvent.HOLD
# All values (2x):
HOLD | RESume
```

3.92 VideoCodec

```
# Example value:
value = enums.VideoCodec.H263
# All values (2x):
H263 | H264
```

3.93 VoicePrecondition

```
# Example value:
value = enums.VoicePrecondition.SIMPle
# All values (3x):
SIMPle | WNPRecondit | WPRecondit
```

3.94 VoimState

```
# Example value:
value = enums.VoimState.EST
# All values (5x):
EST | HOLD | REL | RING | UNK
```

CHAPTER

FOUR

REPCAPS

4.1 MeasInstance (Global)

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

4.2 AccPointName

```
# First value:
value = repcap.AccPointName.Nr1
# Range:
Nr1 .. Nr15
# All values (15x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15
```

4.3 Client

```
# First value:
value = repcap.Client.Ix1
# Range:
Ix1 .. Ix8
# All values (8x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
```

4.4 Codec

```
# First value:
value = repcap.Codec.Ix1
# Range:
Ix1 .. Ix10
# All values (10x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10
```

4.5 Dlink

```
# First value:
value = repcap.Dlink.Ix1
# Values (4x):
Ix1 | Ix2 | Ix3 | Ix4
```

4.6 Fltr

```
# First value:
value = repcap.Fltr.Ix1
# Range:
Ix1 .. Ix15
# All values (15x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10 | Ix11 | Ix12 | Ix13 | Ix14 | Ix15
```

4.7 Impairments

```
# First value:
value = repcap.Impairments.Ix1
# Range:
Ix1 .. Ix15
# All values (15x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10 | Ix11 | Ix12 | Ix13 | Ix14 | Ix15
```

4.8 Ims

```
# First value:
value = repcap.Ims.Ix1
# Values (2x):
Ix1 | Ix2
```

4.9 Imsi

```
# First value:
value = repcap.Imsi.Ix1
# Range:
Ix1 .. Ix10
# All values (10x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10
```

4.10 Nat

```
# First value:
value = repcap.Nat.Ix1
# Range:
Ix1 .. Ix8
# All values (8x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
```

4.11 PcscFnc

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

4.12 Profile

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

4.13 Server

```
# First value:
value = repcap.Server.Ix1
# Range:
Ix1 .. Ix8
# All values (8x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
```

4.14 Slot

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

4.15 Subscriber

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

4.16 Trace

```
# First value:
value = repcap.Trace.Ix1
# Range:
Ix1 .. Ix10
# All values (10x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10
```

4.17 UserId

```
# First value:
value = repcap.UserId.Ix1
# Range:
Ix1 .. Ix10
# All values (10x):
Ix1 | Ix2 | Ix3 | Ix4 | Ix5 | Ix6 | Ix7 | Ix8
Ix9 | Ix10
```

4.18 VirtualSubscriber

```
# First value:
value = repcap.VirtualSubscriber.Nr1
# Range:
Nr1 .. Nr20
# All values (20x):
Nr1 | Nr2 | Nr3 | Nr4 | Nr5 | Nr6 | Nr7 | Nr8
Nr9 | Nr10 | Nr11 | Nr12 | Nr13 | Nr14 | Nr15 | Nr16
Nr17 | Nr18 | Nr19 | Nr20
```

CHAPTER

FIVE

EXAMPLES

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

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

from RsCmwBase import *  # install from pypi.org

RsCmwBase.assert_minimum_version('3.7.90.32')
cmw_base = RsCmwBase('TCPIP::10.112.1.116::INSTR', True, False)
print(f'CMW Base IND: {cmw_base.utilities.idn_string}')
print(f'CMW Instrument options:\n{",".join(cmw_base.utilities.instrument_options)}')
cmw_base.utilities.visa_timeout = 5000

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

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

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

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

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


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

(continues on next page)

(continued from previous page)

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

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

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

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

CHAPTER

SIX

INDEX

CHAPTER

SEVEN

RSCMWDAU API STRUCTURE

Global RepCaps

```
driver = RsCmwDau('TCPIP::192.168.2.101::HISLIP')
# MeasInstance range: Inst1 .. Inst4
rc = driver.repcap_measInstance_get()
driver.repcap_measInstance_set(repcap.MeasInstance.Inst1)
```

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

611 total commands, 5 Sub-groups, 0 group commands

Initializes new RsCmwDau session.

Parameter options tokens examples:

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

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

Parameters

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

static assert_minimum_version(*min_version: str*) → None

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

close() → None

Closes the active RsCmwDau session.

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

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

Parameters

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

get_session_handle() → object

Returns the underlying session handle.

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

Finds all the resources defined by the expression

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

Parameters

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

restore_all_repcaps_to_default() → None

Sets all the Group and Global repcaps to their initial values

Subgroups

7.1 Sense

class Sense
Sense commands group definition. 81 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.1 Data

class Data
Data commands group definition. 81 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.1.1 Control

class Control
Control commands group definition. 60 total commands, 12 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.1.1.1 Services

SCPI Commands

```
SENSe:DATA:CONTrol:SERVices:VERSion
```

class Services

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

get_version() → str

```
# SCPI: SENSe:DATA:CONTrol:SERVices:VERSion
value: str = driver.sense.data.control.services.get_version()
```

No command help available

return version_list: No help available

7.1.1.1.2 Udp

SCPI Commands

```
SENSe:DATA:CONTrol:UDP:RESult
SENSe:DATA:CONTrol:UDP:RECeive
```

class Udp

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

class ReceiveStruct

Structure for reading output parameters. Fields:

- Receive_Status: enums.ReceiveStatusA: No parameter help available
- Timestamp: int: No parameter help available
- Src_Addr: str: No parameter help available
- Src_Port: int: No parameter help available
- Dst_Addr: str: No parameter help available
- Dst_Port: int: No parameter help available
- Packet: List[int]: No parameter help available

class ResultStruct

Structure for reading output parameters. Fields:

- Result: enums.Result: No parameter help available
- Timestamp: int: No parameter help available
- Message: str: No parameter help available

get_receive() → ReceiveStruct

```
# SCPI: SENSe:DATA:CONTrol:UDP:RECeive
value: ReceiveStruct = driver.sense.data.control.udp.get_receive()
```

No command help available

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

get_result() → ResultStruct

```
# SCPI: SENSe:DATA:CONTrol:UDP:RESult
value: ResultStruct = driver.sense.data.control.udp.get_result()
```

No command help available

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

7.1.1.1.3 Deploy

SCPI Commands

```
SENSe:DATA:CONTrol:DEPLoy:RESult
```

class Deploy

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

class ResultStruct

Structure for reading output parameters. Fields:

- Msg_Type: int: No parameter help available
- Msg_Code: int: No parameter help available
- Msg_String: str: No parameter help available

get_result() → ResultStruct

```
# SCPI: SENSe:DATA:CONTrol:DEPLoy:RESult
value: ResultStruct = driver.sense.data.control.deploy.get_result()
```

No command help available

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

7.1.1.1.4 Ims<Ims>

RepCap Settings

```
# Range: Ix1 .. Ix2
rc = driver.sense.data.control.ims.repcap_ims_get()
driver.sense.data.control.ims.repcap_ims_set(repcap.Ims.Ix1)
```

class Ims

Ims commands group definition. 28 total commands, 14 Sub-groups, 0 group commands Repeated Capability: Ims, default value after init: Ims.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.clone()
```

Subgroups

7.1.1.1.4.1 Ecall

class Ecall
Ecall commands group definition. 4 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.ecall.clone()
```

Subgroups

7.1.1.1.4.2 Msd

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:ECALl:MSD
```

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

class GetStruct
Response structure. Fields:

- Count: int: No parameter help available
- Msd: List[int]: No parameter help available

get(*call_id: str*, *ims=<Ims.Default: -1>*) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:ECALl:MSD
value: GetStruct = driver.sense.data.control.ims.ecall.msd.get(call_id = '1',␣
↪ims = repcap.Ims.Default)
```

No command help available

param call_id No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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.sense.data.control.ims.ecall.msd.clone()
```

Subgroups

7.1.1.1.4.3 Extended

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:ECALl:MSD:EXTended
```

class Extended

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

class GetStruct

Response structure. Fields:

- Count: int: No parameter help available
- Content_Type: str: No parameter help available
- Msd: List[int]: No parameter help available

get(*call_id: str*, *ims=<Ims.Default: -1>*) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:ECALl:MSD:EXTended
value: GetStruct = driver.sense.data.control.ims.ecall.msd.extended.get(call_id␣
↪= '1', ims = repcap.Ims.Default)
```

No command help available

param call_id No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.1.1.1.4.4 TypePy

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:ECALl:TYPE
```

class TypePy

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

class GetStruct

Response structure. Fields:

- Type_Py: enums.EcallType: No parameter help available
- Testcall: enums.Testcall: No parameter help available

- Urn: str: No parameter help available

get(*call_id: str*, *ims=<Ims.Default: -1>*) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:ECALl:TYPE
value: GetStruct = driver.sense.data.control.ims.ecall.typePy.get(call_id = '1',
↪ ims = repcap.Ims.Default)
```

No command help available

param call_id No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.1.1.1.4.5 CallId

class CallId

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.ecall.callId.clone()
```

Subgroups

7.1.1.1.4.6 All

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:ECALl:CALLid:ALL
```

class All

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

get(*ims=<Ims.Default: -1>*) → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:ECALl:CALLid:ALL
value: List[str] = driver.sense.data.control.ims.ecall.callId.all.get(ims =␣
↪repcap.Ims.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return nge_call_ids: No help available

7.1.1.1.4.7 Rcs

class Rcs

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.rcs.clone()
```

Subgroups

7.1.1.1.4.8 Participant

class Participant

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.rcs.participant.clone()
```

Subgroups

7.1.1.1.4.9 ListPy

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:RCS:PARTicipant:LIST
```

class ListPy

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

get(*ims=<Ims.Default: -1>*) → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:RCS:PARTicipant:LIST
value: List[str] = driver.sense.data.control.ims.rcs.participant.listPy.get(ims␣
↪= repcap.Ims.Default)
```

Queries the list of participants for group chats.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return list_py: Comma-separated list of strings, one string per participant

7.1.1.1.4.10 Conference

class Conference

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.conference.clone()
```

Subgroups

7.1.1.1.4.11 Factory

class Factory

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.conference.factory.clone()
```

Subgroups

7.1.1.1.4.12 ListPy

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:CONFerence:FACTory:LIST
```

class ListPy

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

get(*ims=<Ims.Default: -1>*) → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:CONFerence:FACTory:LIST
value: List[str] = driver.sense.data.control.ims.conference.factory.listPy.
↪get(ims = repcap.Ims.Default)
```

Queries the factory list (list of reachable conference server addresses) .

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return list_py: Comma-separated list of strings, one string per address

7.1.1.1.4.13 Mobile<Profile>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.sense.data.control.ims.mobile.repcap_profile_get()
driver.sense.data.control.ims.mobile.repcap_profile_set(repcap.Profile.Nr1)
```

SCPI Commands

```
SENSe:DATA:CONTrol:IMS:MOBile:HDOMain
SENSe:DATA:CONTrol:IMS:MOBile:IPADdress
```

class Mobile

Mobile commands group definition. 6 total commands, 3 Sub-groups, 2 group commands Repeated Capability: Profile, default value after init: Profile.Nr1

get_hdomain() → str

```
# SCPI: SENSe:DATA:CONTrol:IMS:MOBile:HDOMain
value: str = driver.sense.data.control.ims.mobile.get_hdomain()
```

No command help available

return home_domain: No help available

get_ip_address() → str

```
# SCPI: SENSe:DATA:CONTrol:IMS:MOBile:IPADdress
value: str = driver.sense.data.control.ims.mobile.get_ip_address()
```

No command help available

return ip_address: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.mobile.clone()
```

Subgroups

7.1.1.1.4.14 CipAddress

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:MOBile<Profile>:CIPaddress
```

class CipAddress

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

get(*ims=<Ims.Default: -1>*, *profile=<Profile.Default: -1>*) → str

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:MOBile<UE>:CIPaddress
value: str = driver.sense.data.control.ims.mobile.cipAddress.get(ims = repcap.
↪Ims.Default, profile = repcap.Profile.Default)
```

Queries the IP addresses of a subscriber.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return ip_address: IP addresses as string

7.1.1.1.4.15 Status

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:MOBile<Profile>:STATus
```

class Status

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

get(*ims=<Ims.Default: -1>*, *profile=<Profile.Default: -1>*) → RsCmwDau.enums.MobileStatus

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:MOBile<UE>:STATus
value: enums.MobileStatus = driver.sense.data.control.ims.mobile.status.get(ims␣
↪= repcap.Ims.Default, profile = repcap.Profile.Default)
```

Queries the state of a subscriber.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return status: UNRegistered | REGistered | EMERgency | EXPired DUT unregistered, registered, emergency registered, registration expired

7.1.1.1.4.16 Uid

SCPI Commands

```
SENSe:DATA:CONTrol:IMS:MOBile:UID:PRIVate
SENSe:DATA:CONTrol:IMS:MOBile:UID:PUBLic
```

class Uid

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

get_private() → str

```
# SCPI: SENSe:DATA:CONTrol:IMS:MOBile:UID:PRIVate
value: str = driver.sense.data.control.ims.mobile.uid.get_private()
```

No command help available

return priv_user_id: No help available

get_public() → str

```
# SCPI: SENSe:DATA:CONTrol:IMS:MOBile:UID:PUBLic
value: str = driver.sense.data.control.ims.mobile.uid.get_public()
```

No command help available

return publ_user_id: No help available

7.1.1.1.4.17 Pcscf

SCPI Commands

```
SENSe:DATA:CONTrol:IMS:PCSCf:STATus
```

class Pcscf

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

get_status() → RsCmwDau.enums.PcScfStatus

```
# SCPI: SENSe:DATA:CONTrol:IMS:PCSCf:STATus
value: enums.PcScfStatus = driver.sense.data.control.ims.pcscf.get_status()
```

No command help available

return status: No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.pcscf.clone()
```

Subgroups

7.1.1.1.4.18 Catalog

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:PCSCf:CATalog
```

class Catalog

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

get(*ims*=<*Ims.Default: -1*>) → str

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:PCSCf:CATalog
value: str = driver.sense.data.control.ims.pcscf.catalog.get(ims = repcap.Ims.
↪Default)
```

Queries a list of all P-CSCF profile names.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return catalog: String listing all profile names, separated by commas String example: 'P-CSCF 1,P-CSCF 2'

7.1.1.1.4.19 Intern

class Intern

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.intern.clone()
```

Subgroups

7.1.1.1.4.20 Pcscf

SCPI Commands

```
SENSe:DATA:CONTrol:IMS:INTern:PCSCf:ADDRess
```

class Pcscf

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

get_address() → str

```
# SCPI: SENSe:DATA:CONTrol:IMS:INTern:PCSCf:ADDRess
value: str = driver.sense.data.control.ims.intern.pcscf.get_address()
```

No command help available

return address: No help available

7.1.1.1.4.21 Ginfo

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:GINFo
```

class Ginfo

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

class GetStruct

Response structure. Fields:

- Timestamp: List[str]: Timestamp of the entry as string in the format 'hh:mm:ss'
- Info_Type: List[enums.InfoType]: NONE | INFO | WARNing | ERRor Category of the entry NONE means that no category is assigned. If no entry at all is available, the answer is '',NONE,".
- Generic_Info: List[str]: Text string describing the event

get(*ims*=<*Ims.Default: -1*>) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:GINFo
value: GetStruct = driver.sense.data.control.ims.ginfo.get(ims = repcap.Ims.
↪Default)
```

Queries all entries of the 'General IMS Info' area. For each entry, three parameters are returned, from oldest to latest entry: {<Timestamp>, <InfoType>, <GenericInfo>}entry 1, {<Timestamp>, <InfoType>, <GenericInfo>}entry 2, ...

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.1.1.1.4.22 VirtualSubscriber<VirtualSubscriber>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.sense.data.control.ims.virtualSubscriber.repcap_virtualSubscriber_get()
driver.sense.data.control.ims.virtualSubscriber.repcap_virtualSubscriber_set(repcap.
↪VirtualSubscriber.Nr1)
```

class VirtualSubscriber
VirtualSubscriber commands group definition. 4 total commands, 4 Sub-groups, 0 group commands Repeated Capability: VirtualSubscriber, default value after init: VirtualSubscriber.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.virtualSubscriber.clone()
```

Subgroups

7.1.1.1.4.23 MtFileTfr

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.virtualSubscriber.mtFileTfr.clone()
```

Subgroups

7.1.1.1.4.24 Destination

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.virtualSubscriber.mtFileTfr.destination.clone()
```

Subgroups

7.1.1.1.4.25 ListPy

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTFiletfr:DESTination:LIST
```

class ListPy

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:DESTination:LIST
value: List[str] = driver.sense.data.control.ims.virtualSubscriber.mtFileTfr.
↪destination.listPy.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Queries a list of all possible destination strings for file transfers by virtual subscriber <v>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return destinations: Comma-separated list of strings, one per destination

7.1.1.1.4.26 MtSms

class MtSms

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.virtualSubscriber.mtSms.clone()
```

Subgroups

7.1.1.1.4.27 Destination

class Destination

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.virtualSubscriber.mtSms.destination.clone()
```

Subgroups

7.1.1.1.4.28 ListPy

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTSMs:DESTination:LIST
```

class ListPy

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:DESTination:LIST
value: List[str] = driver.sense.data.control.ims.virtualSubscriber.mtSms.
↪destination.listPy.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Queries a list of all possible destination strings for mobile-terminating messages.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return destinations: Comma-separated list of strings, one per destination

7.1.1.1.4.29 MtCall

class MtCall

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.virtualSubscriber.mtCall.clone()
```

Subgroups

7.1.1.1.4.30 Destination

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.virtualSubscriber.mtCall.destination.clone()
```

Subgroups

7.1.1.1.4.31 ListPy

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:DESTination:LIST
```

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:DESTination:LIST
value: List[str] = driver.sense.data.control.ims.virtualSubscriber.mtCall.
↪destination.listPy.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Queries a list of all possible destination strings for mobile-terminating calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return destinations: Comma-separated list of strings, one per destination

7.1.1.1.4.32 Catalog

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:VIRTualsub:CATalog
```

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

get(*ims=<Ims.Default: -1>*) → str

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:VIRTualsub:CATalog
value: str = driver.sense.data.control.ims.virtualSubscriber.catalog.get(ims =␣
↪repcap.Ims.Default)
```

Queries a list of all virtual subscriber profile names.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return catalog: String listing all profile names, separated by commas String example: 'Virtual 1,Virtual 2'

7.1.1.1.4.33 Events

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:EVENts
```

class Events

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

class GetStruct

Response structure. Fields:

- Idn: List[str]: String identifying the event log entry Use this ID to query event details via [CMDLINK: SENSe:DATA:CONTrol:IMS2:HISTory CMDLINK]
- Timestamps: List[str]: Timestamp as string in the format 'hh:mm:ss'
- Source: List[str]: Originating party as string
- Destination: List[str]: Terminating party as string
- Type_Py: List[enums.DataType]: AUDio | VIDeo | SMS | INValid | CALL | RCSLmsg | FILetransfer | FTLMode AUDio: audio call VIDeo: video call SMS: sent or received short message CALL: call setup, released call or call on hold RCSLmsg: sent or received RCS large message FILetransfer: file transfer FTLMode: file transfer large mode
- State: List[enums.SessionState]: OK | NOK | PROGgres | RINGing | ESTablished | HOLD | RESumed | RELeased | MEDiaupdate | BUSY | DECLined | INITialmedia | FILetransfer | SRVCcrelease | TERMinated | CANCeled | REJected | CREated Status of the session or message transfer

get(*ims=<Ims.Default: -1>*) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:EVENts
value: GetStruct = driver.sense.data.control.ims.events.get(ims = repcap.Ims.
↪Default)
```

Queries all entries of the event log. For each entry, six parameters are returned: {<ID>, <Timestamps>, <Source>, <Destination>, <Type>, <State>}entry 1, {<ID>, ..., <State>}entry 2, ...

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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.sense.data.control.ims.events.clone()
```

Subgroups

7.1.1.1.4.34 Last

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:EVENts:LAST
```

class Last

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

class GetStruct

Response structure. Fields:

- Idn: str: String identifying the event log entry Use this ID to query event details via [CMDLINK: SENSe:DATA:CONTrol:IMS2:HISTory CMDLINK]
- Timestamps: str: Timestamp as string in the format 'hh:mm:ss'
- Source: str: Originating party as string
- Destination: str: Terminating party as string
- Type_Py: enums.DataType: AUDio | VIDeo | SMS | INValid | CALL | RCSLmsg | FILetransfer | FTLMode AUDio: audio call VIDeo: video call SMS: sent or received short message CALL: call setup, released call or call on hold RCSLmsg: sent or received RCS large message FILetransfer: file transfer FTLMode: file transfer large mode
- State: enums.SessionState: OK | NOK | PROGgres | RINGing | ESTablished | HOLD | RESumed | RELeased | MEDiaupdate | BUSY | DECLined | INITialmedia | FILetransfer | SRVCcrelease | TERMinated | CANCeled | REJected | CREated Status of the session or message transfer

get(*ims*=<*Ims.Default: -1*>) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:EVENts:LAST
value: GetStruct = driver.sense.data.control.ims.events.last.get(ims = repcap.
↪Ims.Default)
```

Queries the last entry of the event log.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.1.1.1.4.35 Subscriber

class Subscriber

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.subscriber.clone()
```

Subgroups

7.1.1.1.4.36 Catalog

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:SUBScriber:CATalog
```

class Catalog

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

get(*ims=<Ims.Default: -1>*) → str

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:SUBScriber:CATalog
value: str = driver.sense.data.control.ims.subscriber.catalog.get(ims = repcap.
↪Ims.Default)
```

Queries a list of all subscriber profile names.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return catalog: String listing all profile names, separated by commas String example: 'Subscriber 1,Subscriber 2'

7.1.1.1.4.37 History

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:HISTory
```

class History

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

class GetStruct

Response structure. Fields:

- Sms_Timestamps: str: Timestamp of the message transfer as string in the format 'hh:mm:ss'
- Sms_Type: enums.SmsTypeA: TGPP | TGPP2 | OGPP | OGPP2 | OPAGer | TPAGer TGPP: mobile-terminating message, 3GPP TGPP2: mobile-terminating message, 3GPP2 OGPP: mobile-originating

message, 3GPP OGPP2: mobile-originating message, 3GPP2 OPAGer: mobile-originating message, RCS pager mode TPAGer: mobile-terminating message, RCS pager mode

- Sms_Encoding: enums.SmsEncoding: GSM7 | GSM8 | UCS | ASCI | IAF5 | NENC | BASE64 Encoding of the message
- Sms_Text: str: Message text as string
- History_State: enums.SessionState: OK | NOK | PROGgres | RINGing | ESTablished | HOLD | RESumed | RELeased | MEDiaupdate | BUSY | DECLined | RCSTxt | INITialmedia | FILetransfer | SRVCcrelease | TERMinated | CANCeled | REJected Status of the call
- History_Timestamps: str: Timestamp of the call as string in the format 'hh:mm:ss'
- Signaling_Type: enums.SignalingType: PRECondit | NOPRecondit | SIMPle | REQU100 | REQuprecondi | WOTPrec183 | EARLymedia Signaling type of the call
- Audio_Codec_Type: enums.CodecType: NARRowband | WIDeband | EVS Audio codec type of the call
- Amr_Align_Mode: enums.AlignMode: OCTetaligned | BANDwidtheff AMR voice codec alignment mode of the call
- Amr_Mode: str: Codec mode as string
- Video_Codec: enums.VideoCodec: H263 | H264 Video codec of the video call
- Video_Attributes: str: Video codec attributes of the video call

`get`(*idn: str*, *ims=<Ims.Default: -1>*) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:HISTory
value: GetStruct = driver.sense.data.control.ims.history.get(idn = '1', ims =␣
↪repcap.Ims.Default)
```

Queries details for a selected event log entry. INTRO_CMD_HELP: The returned sequence depends on the type of the entry. Examples:

- Four values are returned for a message entry of the type 3GPP, 3GPP2 or RCS pager mode: <SMSTimestamps>, <SMSType>, <SMSEncoding>, <SMSText>
- Eight values are returned for each recorded state of a call entry: {<HistoryState>, <HistoryTimestamps>, <SignalingType>, <AudioCodecType>, <AMRAlignMode>, <AMRMode>, <VideoCodec>, <VideoAttributes>}state 1, {...}state 2, ..., {...}state n If a parameter is not relevant for a state, INV is returned for this parameter.

param idn String selecting the event log entry To query IDs, see method **RsCmwDau.Sense.Data.Control.Ims.Events.get_**.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.1.1.1.4.38 Release

class Release

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.release.clone()
```

Subgroups

7.1.1.1.4.39 ListPy

SCPI Commands

```
SENSe:DATA:CONTrol:IMS<Ims>:RELease:LIST
```

class ListPy

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

get(*ims=<Ims.Default: -1>*) → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IMS<Suffix>:RELease:LIST
value: List[str] = driver.sense.data.control.ims.release.listPy.get(ims =␣
↪repcap.Ims.Default)
```

Queries the IDs of all established calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return ids: Comma-separated list of ID strings, one string per established call

7.1.1.1.4.40 Sms

SCPI Commands

```
SENSe:DATA:CONTrol:IMS:SMS:RECeived
```

class Sms

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

class ReceivedStruct

Structure for reading output parameters. Fields:

- Sms_Type: enums.SmsTypeB: No parameter help available
- Sms_Text: str: No parameter help available

get_received() → ReceivedStruct

```
# SCPI: SENSe:DATA:CONTrol:IMS:SMS:RECeived
value: ReceivedStruct = driver.sense.data.control.ims.sms.get_received()
```

No command help available

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.sms.clone()
```

Subgroups

7.1.1.1.4.41 Send

SCPI Commands

```
SENSe:DATA:CONTrol:IMS:SMS:SEND:STATus
```

class Send

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

get_status() → RsCmwDau.enums.SmsStatus

```
# SCPI: SENSe:DATA:CONTrol:IMS:SMS:SEND:STATus
value: enums.SmsStatus = driver.sense.data.control.ims.sms.send.get_status()
```

No command help available

return sms_status: No help available

7.1.1.1.4.42 Voice

class Voice

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ims.voice.clone()
```

Subgroups

7.1.1.1.4.43 Call

SCPI Commands

```
SENSe:DATA:CONTrol:IMS:VOICe:CALL:STATe
```

class Call

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

get_state() → RsCmwDau.enums.CallState

```
# SCPI: SENSe:DATA:CONTrol:IMS:VOICe:CALL:STATe
value: enums.CallState = driver.sense.data.control.ims.voice.call.get_state()
```

No command help available

return call_state: No help available

7.1.1.1.5 Supl

SCPI Commands

```
SENSe:DATA:CONTrol:SUPL:RECeive
SENSe:DATA:CONTrol:SUPL:IFACe
```

class Supl

Supl commands group definition. 3 total commands, 1 Sub-groups, 2 group commands

class ReceiveStruct

Structure for reading output parameters. Fields:

- Receive_Status: enums.ReceiveStatusB: No parameter help available
- Timestamp: int: No parameter help available
- Message: List[int]: No parameter help available

get_iface() → str

```
# SCPI: SENSe:DATA:CONTrol:SUPL:IFACe
value: str = driver.sense.data.control.supl.get_iface()
```

No command help available

return interface_version: No help available

get_receive() → ReceiveStruct

```
# SCPI: SENSe:DATA:CONTrol:SUPL:RECeive
value: ReceiveStruct = driver.sense.data.control.supl.get_receive()
```

No command help available

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.supl.clone()
```

Subgroups

7.1.1.1.5.1 Transmit

SCPI Commands

```
SENSe:DATA:CONTrol:SUPL:TRANsmit:STATus
```

class Transmit

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

class StatusStruct

Structure for reading output parameters. Fields:

- Delivery_Status: enums.ReceiveStatusB: No parameter help available
- Timestamp: int: No parameter help available
- Message: str: No parameter help available

get_status() → StatusStruct

```
# SCPI: SENSe:DATA:CONTrol:SUPL:TRANsmit:STATus
value: StatusStruct = driver.sense.data.control.supl.transmit.get_status()
```

No command help available

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

7.1.1.1.6 Lan

class Lan

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.lan.clone()
```

Subgroups

7.1.1.1.6.1 Dau

SCPI Commands

```
SENSe:DATA:CONTrol:LAN:DAU:STATus
```

class Dau

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

get_status() → RsCmwDau.enums.DauStatus

```
# SCPI: SENSe:DATA:CONTrol:LAN:DAU:STATus
value: enums.DauStatus = driver.sense.data.control.lan.dau.get_status()
```

Queries the state of the LAN DAU connector.

return status: NOTConn | CONN NOTConn: no external network connected CONN: active external network connected

7.1.1.1.7 IpvFour

class IpvFour

IpvFour commands group definition. 6 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvFour.clone()
```

Subgroups

7.1.1.1.7.1 Current

SCPI Commands

```
SENSe:DATA:CONTrol:IPVFour:CURRent:SMASk
SENSe:DATA:CONTrol:IPVFour:CURRent:IPADdress
SENSe:DATA:CONTrol:IPVFour:CURRent:GIP
```

class Current

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

get_gip() → str

```
# SCPI: SENSe:DATA:CONTrol:IPVFour:CURRent:GIP
value: str = driver.sense.data.control.ipvFour.current.get_gip()
```

Queries the current IPv4 address of the gateway server.

return gateway_ip: IP address as string Range: '0.0.0.0' to '255.255.255.255'

get_ip_address() → str

```
# SCPI: SENSe:DATA:CONTrol:IPVFour:CURRent:IPADdress
value: str = driver.sense.data.control.ipvFour.current.get_ip_address()
```

Queries the current IPv4 address of the DAU.

return ip_address: IP address as string Range: '0.0.0.0' to '255.255.255.255'

get_smask() → str

```
# SCPI: SENSe:DATA:CONTrol:IPVFour:CURRent:SMASk
value: str = driver.sense.data.control.ipvFour.current.get_smask()
```

Queries the subnet mask of the current IPv4 data testing configuration.

return subnet_mask: Subnet mask as string Range: '0.0.0.0' to '255.255.255.255'

7.1.1.1.7.2 Static

class Static

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvFour.static.clone()
```

Subgroups

7.1.1.1.7.3 Addresses

SCPI Commands

```
SENSe:DATA:CONTrol:IPVFour:STATic:ADDResses:CATalog
```

class Addresses

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

get_catalog() → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IPVFour:STATic:ADDResses:CATalog
value: List[str] = driver.sense.data.control.ipvFour.static.addresses.get_
↪catalog()
```

Queries the current IPv4 address pool for DUTs, configured statically.

return ip_address: Comma-separated list of strings. Each string represents a static mobile IPv4 address.

7.1.1.1.7.4 Dhcp

class Dhcp

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvFour.dhcp.clone()
```

Subgroups

7.1.1.1.7.5 Addresses

SCPI Commands

```
SENSe:DATA:CONTrol:IPVFour:DHCP:ADDResses:CATalog
```

class Addresses

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

get_catalog() → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IPVFour:DHCP:ADDResses:CATalog
value: List[str] = driver.sense.data.control.ipvFour.dhcp.addresses.get_
↪catalog()
```

Queries the current IPv4 address pool for DUTs, configured via DHCPv4.

return ip_address: Comma-separated list of strings. Each string represents an IPv4 address.

7.1.1.1.7.6 Automatic

class Automatic

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvFour.automatic.clone()
```

Subgroups

7.1.1.1.7.7 Addresses

SCPI Commands

```
SENSe:DATA:CONTrol:IPVFour:AUTomatic:ADDResses:CATalog
```

class Addresses

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

get_catalog() → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IPVFour:AUTomatic:ADDResses:CATalog
value: List[str] = driver.sense.data.control.ipvFour.automatic.addresses.get_
↪catalog()
```

Queries the current IPv4 address pool for DUTs, configured automatically (standalone setup) .

return ip_address: Comma-separated list of strings. Each string represents an IPv4 address.

7.1.1.1.8 IpvSix

class IpvSix

IpvSix commands group definition. 6 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvSix.clone()
```

Subgroups

7.1.1.1.8.1 Current

SCPI Commands

```
SENSe:DATA:CONTrol:IPVSix:CURRent:IPADdress
SENSe:DATA:CONTrol:IPVSix:CURRent:DROuter
```

class Current

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

get_drouter() → str

```
# SCPI: SENSe:DATA:CONTrol:IPVSix:CURRent:DROuter
value: str = driver.sense.data.control.ipvSix.current.get_drouter()
```

Queries the IPv6 address currently used to address the default router.

return def_router: IPv6 address as string, e.g. 'fcb1:abab:cdcd:efe0::1'

get_ip_address() → str

```
# SCPI: SENSe:DATA:CONTrol:IPVSix:CURRent:IPADdress
value: str = driver.sense.data.control.ipvSix.current.get_ip_address()
```

Queries the current IPv6 address of the DAU.

return ip_address: IPv6 address as string, e.g.'fcb1:abab:cdcd:efe0::1/64'

7.1.1.1.8.2 Automatic

class Automatic

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvSix.automatic.clone()
```

Subgroups

7.1.1.1.8.3 Prefixes

SCPI Commands

```
SENSe:DATA:CONTrol:IPVSix:AUTomatic:PREFixes:CATalog
```

class Prefixes

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

get_catalog() → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IPVSix:AUTomatic:PREFixes:CATalog
value: List[str] = driver.sense.data.control.ipvSix.automatic.prefixes.get_
↪catalog()
```

Queries the current IPv6 prefix pool for DUTs, configured automatically (standalone setup) .

return prefixes: Comma-separated list of strings, each string representing a pool entry

7.1.1.1.8.4 Static

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvSix.static.clone()
```

Subgroups

7.1.1.1.8.5 Prefixes

SCPI Commands

```
SENSe:DATA:CONTrol:IPVSix:STATic:PREFixes:CATalog
```

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

get_catalog() → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IPVSix:STATic:PREFixes:CATalog
value: List[str] = driver.sense.data.control.ipvSix.static.prefixes.get_
↪catalog()
```

Queries the current mobile IPv6 prefix pool configured statically.

return prefixes: Comma-separated list of strings, each string representing a pool entry

7.1.1.1.8.6 Dhcp

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvSix.dhcp.clone()
```

Subgroups

7.1.1.1.8.7 Prefixes

SCPI Commands

```
SENSe:DATA:CONTrol:IPVSix:DHCP:PREFixes:CATalog
```

class Prefixes

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

get_catalog() → List[str]

```
# SCPI: SENSe:DATA:CONTrol:IPVSix:DHCP:PREFixes:CATalog
value: List[str] = driver.sense.data.control.ipvSix.dhcp.prefixes.get_catalog()
```

Queries the current IPv6 prefix pool for DUTs, configured via DHCP prefix delegation.

return prefixes: Comma-separated list of strings, each string representing a pool entry

7.1.1.1.8.8 Manual

class Manual

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ipvSix.manual.clone()
```

Subgroups

7.1.1.1.8.9 Routing

SCPI Commands

```
SENSe:DATA:CONTrol:IPVSix:MANual:ROUTing:CATalog
```

class Routing

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

class CatalogStruct

Structure for reading output parameters. Fields:

- Prefixes: List[str]: String specifying an IPv6 prefix
- Routers: List[str]: IPv6 address of assigned router as string

get_catalog() → CatalogStruct

```
# SCPI: SENSe:DATA:CONTrol:IPVSix:MANual:ROUTing:CATalog
value: CatalogStruct = driver.sense.data.control.ipvSix.manual.routing.get_
↪catalog()
```

Queries the pool of manual routes for IPv6. The two values listed below are returned for each route: {<Prefixes>, <Routers>}entry 0, {<Prefixes>, <Routers>}entry 1, ...

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

7.1.1.1.9 Dns

class Dns

Dns commands group definition. 7 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.dns.clone()
```

Subgroups

7.1.1.1.9.1 Current

class Current

Current commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.dns.current.clone()
```

Subgroups

7.1.1.1.9.2 IpvFour

class IpvFour

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.dns.current.ipvFour.clone()
```

Subgroups

7.1.1.1.9.3 Primary

SCPI Commands

```
SENSe:DATA:CONTrol:DNS:CURRent:IPVFour:PRIMary:ADDRess
```

class Primary

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

get_address() → str

```
# SCPI: SENSe:DATA:CONTrol:DNS:CURRent:IPVFour:PRIMary:ADDRess
value: str = driver.sense.data.control.dns.current.ipvFour.primary.get_address()
```

Queries the IPv4 address sent to the DUT as primary DNS server address.

return cdns_prim_ip_4: IPv4 address as string

7.1.1.1.9.4 Secondary

SCPI Commands

```
SENSe:DATA:CONTrol:DNS:CURRent:IPVFour:SECondary:ADDRess
```

class Secondary

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

get_address() → str

```
# SCPI: SENSe:DATA:CONTrol:DNS:CURRent:IPVFour:SECondary:ADDRess
value: str = driver.sense.data.control.dns.current.ipvFour.secondary.get_
↪address()
```

Queries the IPv4 address sent to the DUT as secondary DNS server address.

return cdns_sec_ip_4: IPv4 address as string

7.1.1.1.9.5 IpvSix

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.dns.current.ipvSix.clone()
```

Subgroups

7.1.1.1.9.6 Primary

SCPI Commands

```
SENSe:DATA:CONTrol:DNS:CURRent:IPVSix:PRIMary:ADDRess
```

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

get_address() → str

```
# SCPI: SENSe:DATA:CONTrol:DNS:CURRent:IPVSix:PRIMary:ADDRess
value: str = driver.sense.data.control.dns.current.ipvSix.primary.get_address()
```

Queries the IPv6 address sent to the DUT as primary DNS server address.

return cdns_prim_ip_6: IPv6 address as string

7.1.1.1.9.7 Secondary

SCPI Commands

```
SENSe:DATA:CONTrol:DNS:CURRent:IPVSix:SECondary:ADDRess
```

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

get_address() → str

```
# SCPI: SENSe:DATA:CONTrol:DNS:CURRent:IPVSix:SECondary:ADDRess
value: str = driver.sense.data.control.dns.current.ipvSix.secondary.get_
↪address()
```

Queries the IPv6 address sent to the DUT as secondary DNS server address.

return cdns_sec_ip_6: IPv6 address as string

7.1.1.1.9.8 Local

SCPI Commands

```
SENSe:DATA:CONTrol:DNS:LOCal:CATalog
```

class Local

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

class CatalogStruct

Structure for reading output parameters. Fields:

- Url: List[str]: String specifying the URL of a domain
- Ip: List[str]: Assigned IPv4 address or IPv6 address as string

get_catalog() → CatalogStruct

```
# SCPI: SENSe:DATA:CONTrol:DNS:LOCal:CATalog
value: CatalogStruct = driver.sense.data.control.dns.local.get_catalog()
```

Queries the entries of the local DNS server database for type A or type AAAA DNS queries. The two values listed below are returned for each database entry: {<Url>, <IP>}entry 0, {<Url>, <IP>}entry 1, ...

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

7.1.1.1.9.9 Aservices

SCPI Commands

```
SENSe:DATA:CONTrol:DNS:ASERvices:CATalog
```

class Aservices

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

class CatalogStruct

Structure for reading output parameters. Fields:

- Name: List[str]: String specifying the service name
- Url: List[str]: String specifying the URL of the domain
- Protocol: List[enums.Protocol]: UDP | TCP
- Port: List[int]: Range: 0 to 65654

get_catalog() → CatalogStruct

```
# SCPI: SENSe:DATA:CONTrol:DNS:ASERvices:CATalog
value: CatalogStruct = driver.sense.data.control.dns.aservices.get_catalog()
```

Queries the entries of the local DNS server database for type SRV DNS queries. The four values listed below are returned for each database entry: {<Name>, <Url>, <Protocol>, <Port>}entry 0, {...}entry 1, ...

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

7.1.1.1.9.10 Test

SCPI Commands

```
SENSe:DATA:CONTrol:DNS:TEST:RESults
```

class Test

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

class ResultsStruct

Structure for reading output parameters. Fields:

- V_4_Savailable: enums.TestResult: NONE | SUCCeded | FAILed Indicates whether the server was reachable via its IPv4 address
- V_4_Ares_Rec: enums.TestResult: NONE | SUCCeded | FAILed Indicates whether a query type A, sent to the IPv4 address was successful
- V_44_Ar_Es_Rec: enums.TestResult: NONE | SUCCeded | FAILed Indicates whether a query type AAAA, sent to the IPv4 address was successful
- V_6_Savailable: enums.TestResult: NONE | SUCCeded | FAILed Indicates whether the server was reachable via its IPv6 address
- V_6_Ares_Rec: enums.TestResult: NONE | SUCCeded | FAILed Indicates whether a query type A, sent to the IPv6 address was successful
- V_64_Ares_Rec: enums.TestResult: NONE | SUCCeded | FAILed Indicates whether a query type AAAA, sent to the IPv6 address was successful

get_results() → ResultsStruct

```
# SCPI: SENSe:DATA:CONTrol:DNS:TEST:RESults
value: ResultsStruct = driver.sense.data.control.dns.test.get_results()
```

Queries the results of a foreign DNS server test. NONE indicates that no result is available (e.g. no test started yet) . 'Successful query' means that the domain could be resolved and the DNS server has returned an IP address.

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

7.1.1.1.10 Ftp

class Ftp

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.ftp.clone()
```

Subgroups

7.1.1.1.10.1 User

SCPI Commands

```
SENSe:DATA:CONTrol:FTP:USER:CATalog
```

class User

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

class CatalogStruct

Structure for reading output parameters. Fields:

- User: List[str]: FTP user name as string
- Delete_Allowed: List[bool]: OFF | ON OFF: delete forbidden ON: delete allowed
- Download_Allowed: List[bool]: OFF | ON OFF: download forbidden ON: download allowed
- Upload_Allowed: List[bool]: OFF | ON OFF: upload forbidden ON: upload allowed

get_catalog() → CatalogStruct

```
# SCPI: SENSe:DATA:CONTrol:FTP:USER:CATalog
value: CatalogStruct = driver.sense.data.control.ftp.user.get_catalog()
```

Queries the existing FTP user accounts and their permissions. The four values listed below are returned for each FTP user: {<User>, <DeleteAllowed>, <DownloadAllowed>, <UploadAllowed>}user 1, {...}user 2, ..., {...}user n.

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

7.1.1.1.11 Http

class Http

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.http.clone()
```

Subgroups

7.1.1.1.11.1 Streaming

SCPI Commands

```
SENSe:DATA:CONTrol:HTTP:STReaming:RESult
```

class Streaming

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

class ResultStruct

Structure for reading output parameters. Fields:

- Tid: int: No parameter help available
- Filename: str: No parameter help available
- Container: str: No parameter help available
- Duration: float: No parameter help available
- Video_Codec: str: No parameter help available
- Video_Data_Rate: float: No parameter help available
- Video_Profile: str: No parameter help available
- Video_Level: str: No parameter help available
- Frame_Numerator: int: No parameter help available
- Frame_Denominator: int: No parameter help available
- Height: int: No parameter help available
- Width: int: No parameter help available
- Channel_Count: int: No parameter help available
- Audio_Codec: str: No parameter help available
- Audio_Data_Rate: float: No parameter help available
- Audio_Sampling_Rate: int: No parameter help available
- Data_Type: str: No parameter help available

get_result() → ResultStruct

```
# SCPI: SENSe:DATA:CONTrol:HTTP:STReaming:RESult
value: ResultStruct = driver.sense.data.control.http.streaming.get_result()
```

No command help available

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

7.1.1.1.12 Epdg

class Epdg
Epdg commands group definition. 3 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.epdg.clone()
```

Subgroups

7.1.1.1.12.1 Event

SCPI Commands

```
SENSe:DATA:CONTrol:EPDG:EVENt:LOG
```

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

class LogStruct
Structure for reading output parameters. Fields:

- Timestamps: List[str]: Timestamp of the entry as string in the format 'hh:mm:ss'
- Type_Py: List[enums.InfoType]: NONE | INFO | WARNing | ERRor Category of the entry NONE means that no category is assigned. If no entry at all is available, the answer is '',NONE,".
- Info: List[str]: Text string describing the event

get_log() → LogStruct

```
# SCPI: SENSe:DATA:CONTrol:EPDG:EVENt:LOG
value: LogStruct = driver.sense.data.control.epdg.event.get_log()
```

Queries all entries of the event log. For each entry, three parameters are returned, from oldest to latest entry: {<Timestamps>, <Type>, <Info>}entry 1, {<Timestamps>, <Type>, <Info>}entry 2, ...

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

7.1.1.1.12.2 Connections

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.epdg.connections.clone()
```

Subgroups

7.1.1.1.12.3 Imsi<Imsi>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.sense.data.control.epdg.connections.imsi.repcap_imsi_get()
driver.sense.data.control.epdg.connections.imsi.repcap_imsi_set(repcap.Imsi.Ix1)
```

SCPI Commands

```
SENSe:DATA:CONTrol:EPDG:CONNections:IMSI
```

class Imsi

Imsi commands group definition. 2 total commands, 1 Sub-groups, 1 group commands Repeated Capability: Imsi, default value after init: Imsi.Ix1

get_value() → List[str]

```
# SCPI: SENSe:DATA:CONTrol:EPDG:CONNections:IMSI
value: List[str] = driver.sense.data.control.epdg.connections.imsi.get_value()
```

Queries all IMSIs for which ePDG connections are established.

return imsi: Comma-separated list of 10 values Each IMSI is returned as a string value. The remaining values are filled with INV.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.control.epdg.connections.imsi.clone()
```

Subgroups

7.1.1.1.12.4 Apn

SCPI Commands

```
SENSe:DATA:CONTrol:EPDG:CONNections:IMSI<Imsi>:APN
```

class Apn

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

class GetStruct

Response structure. Fields:

- Name: List[str]: Access point name (APN) as string
- Ip_V_4: List[str]: IPv4 address as string
- Ip_V_6: List[str]: IPv6 address as string

get(*imsi=<Imsi.Default: -1>*) → GetStruct

```
# SCPI: SENSe:DATA:CONTrol:EPDG:CONNections:IMSI<Suffix>:APN
value: GetStruct = driver.sense.data.control.epdg.connections.imsi.apn.get(imsi␣
↪= repcap.Imsi.Default)
```

Queries the connection list for a selected IMSI. The list contains 15 connections. If there are fewer connections, the remaining entries are filled with INV. Three parameters are returned for each of the 15 connections: {<Name>, <IPv4>, <IPv6>}conn 1, {...}conn 2, ..., {...}conn 15

param imsi optional repeated capability selector. Default value: Ix1 (settable in the interface 'Imsi')

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

7.1.1.2 Measurement

class Measurement

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

Cloning the Group

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

Subgroups

7.1.1.2.1 IpAnalysis

class IpAnalysis

IpAnalysis commands group definition. 14 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.1.1.2.1.1 IpcSecurity

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:APPLications
```

class IpcSecurity

IpcSecurity commands group definition. 4 total commands, 2 Sub-groups, 1 group commands

get_applications() → List[str]

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:APPLications
value: List[str] = driver.sense.data.measurement.ipAnalysis.ipcSecurity.get_
↪applications()
```

Queries the list of applications resulting from the IP packet analysis on the application layer.

return applications: Comma-separated list of strings, one string per application

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.ipcSecurity.clone()
```

Subgroups

7.1.1.2.1.2 Kyword

class Kyword

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.ipcSecurity.kyword.clone()
```

Subgroups

7.1.1.2.1.3 Search

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:KYWord:SEARch:LIST
```

class Search

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

get_list_py() → List[str]

```
# SCPI: SENSe:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:KYWord:SEARch:LIST
value: List[str] = driver.sense.data.measurement.ipAnalysis.ipcSecurity.keyword.
↪search.get_list_py()
```

Queries the keyword list.

return list_py: Comma-separated list of strings, one string per keyword

7.1.1.2.1.4 PrtScan

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:STATus
```

class PrtScan

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

get_status() → bool

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:PRTScan:STATus
value: bool = driver.sense.data.measurement.ipAnalysis.ipcSecurity.prtScan.get_
↪status()
```

Queries whether a port scan is running or not.

return scan_trigger: OFF | ON

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.ipcSecurity.prtScan.clone()
```

Subgroups

7.1.1.2.1.5 Event

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:EVENt:LOG
```

class Event

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

class LogStruct

Structure for reading output parameters. Fields:

- Timestamps: List[str]: Timestamp of the entry as string in the format 'hh:mm:ss'
- Type_Py: List[enums.InfoType]: NONE | INFO | WARNing | ERRor Category of the entry NONE means that no category is assigned. If no entry at all is available, the answer is '',NONE,".
- Info: List[str]: Text string describing the event

get_log() → LogStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:EVENt:LOG
value: LogStruct = driver.sense.data.measurement.ipAnalysis.ipcSecurity.prtScan.
↪event.get_log()
```

Queries all entries of the event log. For each entry, three parameters are returned, from oldest to latest entry: {<Timestamps>, <Type>, <Info>}entry 1, {<Timestamps>, <Type>, <Info>}entry 2, ...

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

7.1.1.2.1.6 Export

class Export

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.export.clone()
```

Subgroups

7.1.1.2.1.7 File

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:EXPort:FILE:PATH
```

class File

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

get_path() → str

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:EXPort:FILE:PATH
value: str = driver.sense.data.measurement.ipAnalysis.export.file.get_path()
```

Queries the path and name of the JSON file used for export of the IP analysis result database.

return path: Path and file name as string

7.1.1.2.1.8 IpConnect

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:IPConnect:STATistics
```

class IpConnect

IpConnect commands group definition. 3 total commands, 2 Sub-groups, 1 group commands

class StatisticsStruct

Structure for reading output parameters. Fields:

- No_Of_Conn: int: Total number of connections
- No_Of_Open_Conn: int: Number of open connections
- No_Of_Closed_Conn: int: Number of closed connections
- Open_Tcpc: int: Number of open TCP connections
- Closed_Tcpc: int: Number of closed TCP connections
- Open_Udpc: int: Number of open UDP connections
- Closed_Udpc: int: Number of closed UDP connections

get_statistics() → StatisticsStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:IPConnect:STATistics
value: StatisticsStruct = driver.sense.data.measurement.ipAnalysis.ipConnect.
↪get_statistics()
```

Queries the statistical information provided by the 'IP Connectivity' view.

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.ipConnect.clone()
```

Subgroups

7.1.1.2.1.9 AflowId

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:IPConnect:AFLowid
```

class AflowId

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

class GetStruct

Response structure. Fields:

- App: str: Application name as string
- Feature: str: Feature name as string (for example: audio, video, SMS)

get(*flow_id: float*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:IPConnect:AFLowid
value: GetStruct = driver.sense.data.measurement.ipAnalysis.ipConnect.aflowId.
↪get(flow_id = 1.0)
```

Queries 'IP Connectivity' results for a specific connection, selected via its flow ID.

param flow_id Selects the connection for which information is queried

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

7.1.1.2.1.10 FlowId

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:IPConnect:FLOWid
```

class FlowId

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

class GetStruct

Response structure. Fields:

- Conn_Status: enums.ConnStatus: OPEN | CLOSed Connection status
- Protocol: str: Layer 4 protocol as string ('TCP', 'UDP', ...)
- Dpi_Protocol: str: Layer 7 protocol as string ('HTTP', 'FTP', ...)
- Ip_Addr_Source: str: IP address of the connection source as string
- Ip_Port_Source: int: Port number of the connection source Range: 0 to 65654

- Ip_Addr_Dest: str: IP address of the connection destination as string
- Ip_Port_Dest: int: Port number of the connection destination Range: 0 to 65654
- Overh_Down: float: Downlink overhead as percentage of the packet Range: 0 % to 100 %, Unit: %
- Overh_Up: float: Uplink overhead as percentage of the packet Range: 0 % to 100 %, Unit: %
- Avg_Ps_Down: float: Average downlink packet size Range: 0 bytes to 65535 bytes, Unit: bytes
- Avg_Ps_Up: float: Average uplink packet size Range: 0 bytes to 65535 bytes, Unit: bytes
- App: str: Application name as string
- Country: str: Country of the destination as string (two-letter country code)

get(*flow_id: float*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:IPConnect:FLOWid
value: GetStruct = driver.sense.data.measurement.ipAnalysis.ipConnect.flowId.
↪get(flow_id = 1.0)
```

Queries the 'IP Connectivity' results for a specific connection, selected via its flow ID.

param flow_id Selects the connection for which information is queried

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

7.1.1.2.1.11 TcpAnalysis

class TcpAnalysis

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.tcpAnalysis.clone()
```

Subgroups

7.1.1.2.1.12 FlowId

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:FLOWid
```

class FlowId

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

class GetStruct

Response structure. Fields:

- Th_Down: float: Throughput in downlink direction Unit: bit/s
- Tcpws_Down: enums.OverhUp: OK | FULL Threshold check result for downlink TCP window size

- Retr_Down: enums.OverhUp: OK | NOK Threshold check result for downlink retransmissions
- Overh_Down: enums.OverhUp: OK | NOK Only for backward compatibility - no longer used
- Th_Up: float: Throughput in uplink direction Unit: bit/s
- Tcpws_Up: enums.OverhUp: OK | FULL Threshold check result for uplink TCP window size
- Retr_Up: enums.OverhUp: OK | NOK Threshold check result for uplink retransmissions
- Overh_Up: enums.OverhUp: OK | NOK Only for backward compatibility - no longer used
- Destination: str: Destination address as string
- Rtt: enums.OverhUp: OK | NOK Threshold check result for round-trip time
- Pkt_Size_Up: int: Layer 3 uplink packet size Unit: byte
- Pkt_Size_Dl: int: Layer 3 downlink packet size Unit: byte

get(*flow_id: float*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:FLOWid
value: GetStruct = driver.sense.data.measurement.ipAnalysis.tcpAnalysis.flowId.
↪get(flow_id = 1.0)
```

Queries the threshold check and throughput results for a specific connection, selected via its flow ID.

param flow_id Selects the connection for which information is queried

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

7.1.1.2.1.13 Details

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:DETails
```

class Details

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

class GetStruct

Response structure. Fields:

- Ip_Addr_Source: str: IP address of the connection source as string
- Ip_Port_Source: int: Port number of the connection source Range: 0 to 65654
- Ip_Addr_Dest: str: IP address of the connection destination as string
- Ip_Port_Dest: int: Port number of the connection destination Range: 0 to 65654
- Curr_Tcpws_Down: float: Measured downlink TCP window size Unit: byte
- Max_Tcpws_Down: float: Negotiated maximum downlink TCP window size Unit: byte
- Retr_Down: float: Downlink retransmission rate Range: 0 % to 100 %, Unit: %
- Overh_Down: float: Only for backward compatibility - no longer used Range: 0 % to 100 %, Unit: %
- Curr_Tcpws_Up: float: Measured uplink TCP window size Unit: byte
- Max_Tcpws_Up: float: Negotiated maximum uplink TCP window size Unit: byte

- Retr_Up: float: Uplink retransmission rate Range: 0 % to 100 %, Unit: %
- Overh_Up: float: Only for backward compatibility - no longer used Range: 0 % to 100 %, Unit: %
- Rtt: int: Round-trip time Unit: ms

get(*flow_id: float*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:DETails
value: GetStruct = driver.sense.data.measurement.ipAnalysis.tcpAnalysis.details.
↪get(flow_id = 1.0)
```

Queries the 'TCP Analysis' details for a specific connection, selected via its flow ID.

param flow_id Selects the connection for which information is queried

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

7.1.1.2.1.14 VoIms

class VoIms

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.voIms.clone()
```

Subgroups

7.1.1.2.1.15 Bitrate

class Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.sense.data.measurement.ipAnalysis.voIms.bitrate.clone()
```

Subgroups

7.1.1.2.1.16 Cmr

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:VOIMs:BITRate:CMR
```

class Cmr

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

class GetStruct

Response structure. Fields:

- Flow_Id: int: Flow ID, as returned by [CMDLINK: FETCh:DATA:MEASi:IPANalysis:VOIMs:ALL CMDLINK]
- Direction: enums.DirectionB: UL | DL | UNK Flow direction uplink, downlink or unknown
- Curr_Bitrate: float: Current measured bitrate Unit: bit/s
- Avg_Bitrate: float: Average measured bitrate Unit: bit/s
- Min_Bitrate: float: Minimum measured bitrate Unit: bit/s
- Max_Bitrate: float: Maximum measured bitrate Unit: bit/s
- Curr_Cmr_Bitrate: float: Bitrate currently requested for the other direction Unit: bit/s
- Min_Cmr_Bitrate: float: Minimum of the bitrates requested for the other direction Unit: bit/s
- Max_Cmr_Bitrate: float: Maximum of the bitrates requested for the other direction Unit: bit/s
- Curr_Cmr_Bw: str: String indicating the bandwidth currently requested for the other direction
- Min_Cmr_Bw: str: String indicating the minimum of the bandwidths requested for the other direction
- Max_Cmr_Bw: str: String indicating the maximum of the bandwidths requested for the other direction

get(*session_id: float*, *flow_id: int*, *direction: RsCmwDau.enums.DirectionB*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:VOIMs:BITRate:CMR
value: GetStruct = driver.sense.data.measurement.ipAnalysis.voIms.bitrate.cmr.
↪get(session_id = 1.0, flow_id = 1, direction = enums.DirectionB.DL)
```

Queries bitrates and CMR information related to a voice over IMS call. A query returns all parameters except the <SessionID>: <FlowID>, <Direction>, <CurrBitrate>, <AvgBitrate>, ..., <MaxCMRBW>

param session_id Call ID, as returned by method RsCmwDau.Data.Measurement.IpAnalysis.VoIms.All.fetch

param flow_id Flow ID, as returned by method RsCmwDau.Data.Measurement.IpAnalysis.VoIms.All.fetch

param direction UL | DL | UNK Flow direction uplink, downlink or unknown

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

7.1.1.2.1.17 Flows

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:VOIMs:FLOWs
```

class Flows

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

class GetStruct

Response structure. Fields:

- Flow_Id: int: Flow ID, as returned by [CMDLINK: FETCh:DATA:MEASi:IPANalysis:VOIMs:ALL CMDLINK]
- Direction: enums.DirectionB: UL | DL | UNK Flow direction uplink, downlink or unknown
- Type_Py: enums.AvTypeB: AUDio | VIDeo | UNKNow Flow type audio, video or unknown
- Codec: str: String indicating the used codec
- Seq_Number: int: Sequence number of the currently processed packet
- Num_Pack: int: Number of already processed packets
- Throughput: float: Current audio or video data throughput at the RTP level Unit: bit/s
- Dest_Port: int: Port used at the flow destination
- Evs_Mode: str: String indicating the EVS mode (primary or AMR-WB-IO)
- Evs_Format: str: String indicating the EVS format (header-full or compact)
- Num_Evs_Comp: int: Number of EVS packets with compact format
- Num_Ev_Shp: int: Number of EVS packets with header-full format
- Video_Resolution: str: String indicating the video resolution
- Video_Frate: int: Video frame rate in frames per second
- Video_Oreintation: str: String indicating the counter-clockwise video rotation
- Video_Profile: str: String indicating the H.264 profile
- Video_Level: str: String indicating the H.264 level
- Video_Constraint: str: String indicating the H.264 constraint set
- Bitrate: float: Audio codec rate

get(*session_id: float*, *flow_id: int*, *direction: RsCmwDau.enums.DirectionB*, *type_py: Optional[RsCmwDau.enums.AvTypeB] = None*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:VOIMs:FLOWs
value: GetStruct = driver.sense.data.measurement.ipAnalysis.voIms.flows.
↪get(session_id = 1.0, flow_id = 1, direction = enums.DirectionB.DL, type_py =␣
↪enums.AvTypeB.AUDio)
```

Queries flow information related to a voice over IMS call. A query returns all parameters except the <SessionID>: <FlowID>, <Direction>, <Type>, <Codec>, <SeqNumber>, ..., <Bitrate>

param session_id Call ID, as returned by method RsCmwDau.Data.Measurement.IpAnalysis.VoIms.All.fetch

param flow_id Flow ID, as returned by method RsCmwDau.Data.Measurement.IpAnalysis.VoIms.All.fetch

param direction UL | DL | UNK Flow direction uplink, downlink or unknown

param type_py AUDio | VIDeo | UNKNow Flow type audio, video or unknown

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

7.1.1.2.1.18 PerdTx

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:VOIMs:PERDtx
```

class PerdTx

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

class GetStruct

Response structure. Fields:

- Per_Up: int: No parameter help available
- Per_Down: int: No parameter help available
- Dtx_Up: int: Discontinuous transmission rate in the uplink Range: 0 % to 100 %, Unit: %
- Dtx_Down: int: Discontinuous transmission rate in the downlink Range: 0 % to 100 %, Unit: %

get(*con_id: float*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:VOIMs:PERDtx
value: GetStruct = driver.sense.data.measurement.ipAnalysis.voIms.perdTx.
↪get(con_id = 1.0)
```

Queries the packets measurement results for a selected voice over IMS call. To get a list of all calls and their IDs, use method RsCmwDau.Data.Measurement.IpAnalysis.VoIms.All.fetch.

param con_id Selects the call for which the results are queried

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

7.1.1.2.1.19 Jitter

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPANalysis:VOIMs:JITTer
```

class Jitter

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

class GetStruct

Response structure. Fields:

- Jitter_Max_Up: float: Maximum jitter in the uplink Unit: s
- Jitter_Avg_Up: float: Average jitter in the uplink Unit: s
- Jitter_Min_Up: float: Minimum jitter in the uplink Unit: s
- Jitter_Max_Down: float: Maximum jitter in the downlink Unit: s
- Jitter_Avg_Down: float: Average jitter in the downlink Unit: s
- Jitter_Min_Down: float: Minimum jitter in the downlink Unit: s
- Jitter_Curr_Up: float: Current jitter in the uplink Unit: s
- Jitter_Curr_Down: float: Current jitter in the downlink Unit: s

get(*con_id: float*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPANalysis:VOIMs:JITTer
value: GetStruct = driver.sense.data.measurement.ipAnalysis.voIms.jitter.
↪get(con_id = 1.0)
```

Queries the jitter results for a selected voice over IMS call. To get a list of all calls and their IDs, use method RsCmwDau.Data.Measurement.IpAnalysis.VoIms.All.fetch.

param con_id Selects the call for which the results are queried

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

7.1.1.2.2 Throughput

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:THRoughput:INTerval
```

class Throughput

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

get_interval() → float

```
# SCPI: SENSe:DATA:MEASurement<Instance>:THRoughput:INTerval
value: float = driver.sense.data.measurement.throughput.get_interval()
```

Queries the time interval between two throughput measurement results.

return interval: In the current software version, the value is fixed. Unit: s

7.1.1.2.3 DnsRequests

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:DNSRequests:RCOunt
SENSe:DATA:MEASurement<MeasInstance>:DNSRequests
```

class DnsRequests

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

class ValueStruct

Structure for reading output parameters. Fields:

- Reliability: int: See 'Reliability Indicator'
- Client_Ip: List[str]: String indicating the IP address of the client (DUT) that has sent the DNS request
- Url: List[str]: String indicating the domain or application to be resolved
- Ip: List[str]: String indicating the IP address or domain returned as answer to the DNS request
- Timestamp: List[str]: Timestamp as string in the format 'hh:mm:ss'

get_rcount() → int

```
# SCPI: SENSe:DATA:MEASurement<Instance>:DNSRequests:RCOunt
value: int = driver.sense.data.measurement.dnsRequests.get_rcount()
```

Queries the number of already monitored DNS requests.

return req_count: Number of requests

get_value() → ValueStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:DNSRequests
value: ValueStruct = driver.sense.data.measurement.dnsRequests.get_value()
```

Queries information about the monitored DNS requests. After the reliability indicator, four results are returned for each DNS request: <Reliability>, {<ClientIP>, <URL>, <IP>, <Timestamp>}request 1, {...}request 2, ... To query the number of monitored requests, see method RsCmwDau.Sense.Data.Measurement.DnsRequests.rcount.

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

7.1.1.2.4 IpLogging

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPLogging:FNAMe
```

class IpLogging

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

get_fname() → str

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPLogging:FNAMe
value: str = driver.sense.data.measurement.ipLogging.get_fname()
```

Queries the current or next log file name. If IP logging is on, the name of the currently used log file is returned. If IP logging is off, the name of the file to be created in the next logging session is returned.

return filename: File name as string

7.1.1.2.5 IpReplay

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPReplay:PROGress
```

class IpReplay

IpReplay commands group definition. 3 total commands, 2 Sub-groups, 1 group commands

class ProgressStruct

Structure for reading output parameters. Fields:

- Filename: List[str]: File name as a string
- Progress: List[int]: Range: 0 % to 100 %, Unit: %

get_progress() → ProgressStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPReplay:PROGress
value: ProgressStruct = driver.sense.data.measurement.ipReplay.get_progress()
```

Queries the replay progress for all files in the playlist. The results are returned as follows: {<FileName>, <Progress>}file 1, {...}file 2, ..., {...}file n

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

Cloning the Group

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

Subgroups

7.1.1.2.5.1 InfoFile

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPReplay:INFofile
```

class InfoFile

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

class GetStruct

Response structure. Fields:

- Number_Of_Packets: int: Number of IP packets in the file
- File_Size: int: Unit: byte
- Bitrate: float: Unit: bit/s
- Duration: int: Unit: s
- Type_Py: str: String indicating the file type and information about the capturing application
- Encapsulation: str: 'Raw IP': file contains raw IP traffic 'Ethernet': file contains IP traffic plus Ethernet headers

get(*filename: str*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPReplay:INFofile
value: GetStruct = driver.sense.data.measurement.ipReplay.infoFile.get(filename␣
↪= '1')
```

Queries information about a selected file in the playlist. If the file has not yet been analyzed and the measurement state is RUN, the command triggers file analysis. The analysis takes some time. Repeat the command until the analysis results are available.

param filename File name as a string. Specify the file name with extension but without path, for example 'myfile.pcap'.

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

7.1.1.2.5.2 TrafficFile

SCPI Commands

```
SENSe:DATA:MEASurement<MeasInstance>:IPReplay:TRAFficfile
```

class TrafficFile

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

class GetStruct

Response structure. Fields:

- L_4_Protocol: List[str]: Layer 4 protocol as string ('TCP', 'UDP', ...)
- No_Of_Packets: List[int]: Number of IP packets for the connection
- Ip_Src_Address: List[str]: IP address of the connection source as string
- Ip_Src_Port: List[int]: Port number of the connection source
- Ip_Dest_Address: List[str]: IP address of the connection destination as string
- Ip_Dest_Port: List[int]: Port number of the connection destination

get(*filename: str*) → GetStruct

```
# SCPI: SENSe:DATA:MEASurement<Instance>:IPReplay:TRAFficfile
value: GetStruct = driver.sense.data.measurement.ipReplay.trafficFile.
↪get(filename = '1')
```

Queries information about all IP connections contained in a selected file of the playlist. If the file has not yet been analyzed and the measurement state is RUN, the command triggers file analysis. The analysis takes some time. Repeat the command until the analysis results are available. The results are returned as follows: {<L4Protocol>, <NoOfPackets>, ..., <IPDstPort>}conn 1, {...}conn 2, ...

param filename File name as a string. Specify the file name with extension but without path, for example 'myfile.pcap'.

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

7.2 Configure

class Configure

Configure commands group definition. 389 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1 Data

class Data

Data commands group definition. 388 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1 Control

SCPI Commands

```
CONFigure:DATA:CONTrol:MTU
```

class Control

Control commands group definition. 271 total commands, 11 Sub-groups, 1 group commands

get_mtu() → int

```
# SCPI: CONFigure:DATA:CONTrol:MTU
value: int = driver.configure.data.control.get_mtu()
```

Specifies the MTU, that is the maximum IP packet size that can be transmitted without fragmentation.

return max_trans_unit: Range: 552 bytes to 4096 bytes, Unit: bytes

set_mtu(*max_trans_unit: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:MTU
driver.configure.data.control.set_mtu(max_trans_unit = 1)
```

Specifies the MTU, that is the maximum IP packet size that can be transmitted without fragmentation.

param max_trans_unit Range: 552 bytes to 4096 bytes, Unit: bytes

Cloning the Group

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

Subgroups

7.2.1.1.1 Udp

SCPI Commands

```
CONFigure:DATA:CONTrol:UDP:CLOSe
```

class Udp

Udp commands group definition. 3 total commands, 2 Sub-groups, 1 group commands

close(*ip_address: str*, *port: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:UDP:CLOSe
driver.configure.data.control.udp.close(ip_address = '1', port = 1)
```

No command help available

param ip_address No help available

param port No help available

Cloning the Group

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

Subgroups

7.2.1.1.1.1 Test

SCPI Commands

```
CONFigure:DATA:CONTrol:UDP:TEST
```

class Test

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

get() → List[int]

```
# SCPI: CONFigure:DATA:CONTrol:UDP:TEST
value: List[int] = driver.configure.data.control.udp.test.get()
```

No command help available

return result: No help available

set(*src_port: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:UDP:TEST
driver.configure.data.control.udp.test.set(src_port = 1)
```

No command help available

param src_port No help available

7.2.1.1.2 Bind

SCPI Commands

```
CONFigure:DATA:CONTrol:UDP:BIND
```

class Bind

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

set(*ip_address: str*, *port: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:UDP:BIND
driver.configure.data.control.udp.bind.set(ip_address = '1', port = 1)
```

No command help available

param ip_address No help available

param port No help available

7.2.1.1.2 Deploy

SCPI Commands

```
CONFigure:DATA:CONTrol:DEPLoy
```

class Deploy

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

set(*app_name: str*, *hash_py: str*, *bin_fil_name: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DEPLoy
driver.configure.data.control.deploy.set(app_name = '1', hash_py = r1, bin_fil_
↪name = '1')
```

No command help available

param app_name No help available

param hash_py No help available

param bin_fil_name No help available

7.2.1.1.3 Ims<Ims>

RepCap Settings

```
# Range: Ix1 .. Ix2
rc = driver.configure.data.control.ims.repcap_ims_get()
driver.configure.data.control.ims.repcap_ims_set(repcap.Ims.Ix1)
```

class Ims

Ims commands group definition. 185 total commands, 19 Sub-groups, 0 group commands Repeated Capability: Ims, default value after init: Ims.Ix1

Cloning the Group

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

Subgroups

7.2.1.1.3.1 VirtualSubscriber<VirtualSubscriber>

RepCap Settings

```
# Range: Nr1 .. Nr20
rc = driver.configure.data.control.ims.virtualSubscriber.repcap_virtualSubscriber_get()
driver.configure.data.control.ims.virtualSubscriber.repcap_virtualSubscriber_set(repcap.
↪VirtualSubscriber.Nr1)
```

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:DELete
```

class VirtualSubscriber

VirtualSubscriber commands group definition. 82 total commands, 24 Sub-groups, 1 group commands Repeated Capability: VirtualSubscriber, default value after init: VirtualSubscriber.Nr1

delete(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:DELete
driver.configure.data.control.ims.virtualSubscriber.delete(ims = repcap.Ims.
↪Default, virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Deletes the virtual subscriber profile number <v>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

delete_with_opc(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

Cloning the Group

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

Subgroups

7.2.1.1.3.2 EcConfig

class EcConfig

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

Cloning the Group

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

Subgroups

7.2.1.1.3.3 Actmypes

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:ECConfig:ACTMypes
```

class Actmypes

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

class SetStruct

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

- Msd_Value_1: str: No parameter help available
- Msd_Value_2: str: No parameter help available
- Msd_Value_3: str: No parameter help available
- Msd_Value_4: str: No parameter help available
- Msd_Value_5: str: No parameter help available
- Msd_Value_6: str: No parameter help available
- Msd_Value_7: str: No parameter help available
- Msd_Value_8: str: No parameter help available
- Msd_Value_9: str: No parameter help available
- Msd_Value_10: str: No parameter help available
- Msd_Value_11: str: No parameter help available
- Msd_Value_12: str: No parameter help available
- Msd_Value_13: str: No parameter help available
- Msd_Value_14: str: No parameter help available
- Msd_Value_15: str: No parameter help available
- Msd_Value_16: str: No parameter help available

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.EcConfig_.Actmypes.Actmypes.SetStruct*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:ECConfig:ACTMypes
driver.configure.data.control.ims.virtualSubscriber.ecConfig.actmypes.set(value␣
↪= [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.4 Acuris

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:ECConfig:ACURis
```

class Acuris

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

class SetStruct

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

- Uri_Value_1: str: No parameter help available
- Uri_Value_2: str: No parameter help available
- Uri_Value_3: str: No parameter help available
- Uri_Value_4: str: No parameter help available
- Uri_Value_5: str: No parameter help available
- Uri_Value_6: str: No parameter help available
- Uri_Value_7: str: No parameter help available
- Uri_Value_8: str: No parameter help available
- Uri_Value_9: str: No parameter help available
- Uri_Value_10: str: No parameter help available
- Uri_Value_11: str: No parameter help available
- Uri_Value_12: str: No parameter help available
- Uri_Value_13: str: No parameter help available
- Uri_Value_14: str: No parameter help available
- Uri_Value_15: str: No parameter help available
- Uri_Value_16: str: No parameter help available

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.EcConfig_.Acuris.Acuris.SetStruct*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:ECConfig:ACURis
driver.configure.data.control.ims.virtualSubscriber.ecConfig.acuris.set(value =␣
↪[PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.5 FwdCall

class FwdCall

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

Cloning the Group

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

Subgroups

7.2.1.1.3.6 After

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:FWDCall:AFTer
```

class After

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:FWDCall:AFTer
value: int = driver.configure.data.control.ims.virtualSubscriber.fwdCall.after.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Configures a ringing time for virtual subscriber behaviors where the phone is ringing before the call is accepted or forwarded.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return fwd_after: Range: 0 s to 999 s, Unit: s

set(*fwd_after: int*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:FWDCall:AFTer
driver.configure.data.control.ims.virtualSubscriber.fwdCall.after.set(fwd_after␣
↪= 1, ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

(continues on next page)

(continued from previous page)

Configures a ringing time for virtual subscriber behaviors where the phone is ringing before the call is accepted or forwarded.

param fwd_after Range: 0 s to 999 s, Unit: s

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.7 Session

class Session

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

Cloning the Group

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

Subgroups

7.2.1.1.3.8 Expiry

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:SESSion:EXPiry
```

class Expiry

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SESSion:EXPiry
value: int = driver.configure.data.control.ims.virtualSubscriber.session.expiry.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Configures the 'Session-Expires' header field for the session expiration timer feature.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return expiry: Range: 0 to 64535

set(*expiry: int*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SESSion:EXPiry
driver.configure.data.control.ims.virtualSubscriber.session.expiry.set(expiry =␣
↪1, ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Configures the 'Session-Expires' header field for the session expiration timer feature.

param expiry Range: 0 to 64535

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.9 MinSe

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:SESSion:MINSe
```

class MinSe

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SESSion:MINSe
value: int = driver.configure.data.control.ims.virtualSubscriber.session.minSe.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Configures the 'Min-SE' header field for the session expiration timer feature.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return min_se: Range: 0 to 64535

set(*min_se: int*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SESSion:MINSe
driver.configure.data.control.ims.virtualSubscriber.session.minSe.set(min_se =␣
↪1, ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Configures the 'Min-SE' header field for the session expiration timer feature.

param min_se Range: 0 to 64535

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.10 Usage

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:SESSion:USAGe
```

class Usage

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.SessionUsage

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SESSion:USAGe
value: enums.SessionUsage = driver.configure.data.control.ims.virtualSubscriber.
↪session.usage.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects in which cases session timers are used.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return usage: OFF | ONALways | ONBYue OFF, ON always, ON if requested by UE

set(*usage: RsCmwDau.enums.SessionUsage*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SESSion:USAGe
driver.configure.data.control.ims.virtualSubscriber.session.usage.set(usage =␣
↪enums.SessionUsage.OFF, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects in which cases session timers are used.

param usage OFF | ONALways | ONBYue OFF, ON always, ON if requested by UE

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.11 Evs

class Evs
Evs commands group definition. 15 total commands, 13 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.12 Io

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.io.clone()
```

Subgroups

7.2.1.1.3.13 Mode

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.io.mode.clone()
```

Subgroups

7.2.1.1.3.14 Config

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:IO:MODE:CONFig
```

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.EvsIoModeCnfg

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:IO:MODE:CONFig
value: enums.EvsIoModeCnfg = driver.configure.data.control.ims.
↪virtualSubscriber.evs.io.mode.config.get(ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the speech packet type to be sent from the audio board to the DUT for the EVS AMR-WB IO mode.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return evs_io_mode_cnfg: EVSamrwb | AMRWb AMRWb: AMR-WB encoded packets EVSamrwb: EVS AMR-WB IO encoded packets

set(*evs_io_mode_cnfg: RsCmwDau.enums.EvsIoModeCnfg*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:IO:MODE:CONFig
driver.configure.data.control.ims.virtualSubscriber.evs.io.mode.config.set(evs_
↪io_mode_cnfg = enums.EvsIoModeCnfg.AMRWb, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the speech packet type to be sent from the audio board to the DUT for the EVS AMR-WB IO mode.

param evs_io_mode_cnfg EVSamrwb | AMRWb AMRWb: AMR-WB encoded packets EVSamrwb: EVS AMR-WB IO encoded packets

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.15 Codec<Codec>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.virtualSubscriber.evs.codec.repcap_codec_get()
driver.configure.data.control.ims.virtualSubscriber.evs.codec.repcap_codec_set(repcap.
↪Codec.Ix1)
```

class Codec

Codec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Codec, default value after init: Codec.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.codec.clone()
```

Subgroups

7.2.1.1.3.16 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:CODec<Codec>:ENABle
```

class Enable

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:CODec<CodecIdx>:ENABle
value: bool = driver.configure.data.control.ims.virtualSubscriber.evs.codec.
↪enable.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

Enables or disables a codec rate for the AMR-WB IO mode of the EVS codec.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

return codec_rate: OFF | ON OFF: codec rate not supported ON: codec rate supported

set(*codec_rate: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:CODec<CodecIdx>:ENABle
driver.configure.data.control.ims.virtualSubscriber.evs.codec.enable.set(codec_
↪rate = False, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

Enables or disables a codec rate for the AMR-WB IO mode of the EVS codec.

param codec_rate OFF | ON OFF: codec rate not supported ON: codec rate supported

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

7.2.1.1.3.17 Common

class Common

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.common.clone()
```

Subgroups

7.2.1.1.3.18 Bitrate

class Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.common.bitrate.clone()
```

Subgroups

7.2.1.1.3.19 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:COMMon:BITRate:RANGe
```

class Range

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

class RangeStruct

Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Lower end of the range, 5.9 kbit/s to 128 kbit/s
- Bitrate_Higher: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Upper end of the range, 5.9 kbit/s to 128 kbit/s

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:COMMon:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.virtualSubscriber.evs.
↪common.bitrate.range.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the bit-rate range supported in the EVS primary mode. The value must be compatible to the selected bandwidth. The setting applies only if the uplink (receive) and the downlink (send) are configured together, see method RsCmwDau. Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.Evs_.Common_.Bitrate_.Range.Range.RangeStr*
ims=<Ims.Default: -1>, virtualSubscriber=<VirtualSubscriber.Default: -1>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:COMMon:BITRate:RANGe
driver.configure.data.control.ims.virtualSubscriber.evs.common.bitrate.range.
↪set(value = [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the bit-rate range supported in the EVS primary mode. The value must be compatible to the selected bandwidth. The setting applies only if the uplink (receive) and the downlink (send) are configured together, see method RsCmwDau. Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.20 Receive

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.receive.clone()
```

Subgroups

7.2.1.1.3.21 Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.receive.bitrate.clone()
```

Subgroups

7.2.1.1.3.22 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:RECeive:BITRate:RANGe
```

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

class RangeStruct
Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Lower end of the range, 5.9 kbit/s to 128 kbit/s
- Bitrate_Higher: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Upper end of the range, 5.9 kbit/s to 128 kbit/s

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
:EVS:RECeive:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.virtualSubscriber.evs.
receive.bitrate.range.get(ims = repcap.Ims.Default, virtualSubscriber =␣
repcap.VirtualSubscriber.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the uplink (receive) direction. The value must be compatible to the selected bandwidth. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.Evs_.Receive_.Bitrate_.Range.Range.RangeStruc*
ims=<Ims.Default: -1>, virtualSubscriber=<VirtualSubscriber.Default: -1>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:RECeive:BITRate:RANGe
driver.configure.data.control.ims.virtualSubscriber.evs.receive.bitrate.range.
↪set(value = [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the uplink (receive) direction. The value must be compatible to the selected bandwidth. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.23 Bw

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:RECeive:BW
```

class Bw

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

get(*ims=<Ims.Default: -1>, virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:RECeive:BW
value: enums.Bandwidth = driver.configure.data.control.ims.virtualSubscriber.
↪evs.receive.bw.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the uplink (receive) direction. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims. VirtualSubscriber.Evs.Synch.Select.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return rx_bw: NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

set(*rx_bw: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:RECeive:BW
driver.configure.data.control.ims.virtualSubscriber.evs.receive.bw.set(rx_bw =␣
↪enums.Bandwidth.FB, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the uplink (receive) direction. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims. VirtualSubscriber.Evs.Synch.Select.set.

param rx_bw NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.24 Send

class Send

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.send.clone()
```

Subgroups

7.2.1.1.3.25 Bw

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:SEND:BW
```

class Bw

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:SEND:BW
value: enums.Bandwidth = driver.configure.data.control.ims.virtualSubscriber.
↪evs.send.bw.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the downlink (send) direction. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims. VirtualSubscriber.Evs.Synch.Select.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return send_bw: NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

set(*send_bw: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:SEND:BW
driver.configure.data.control.ims.virtualSubscriber.evs.send.bw.set(send_bw =␣
↪enums.Bandwidth.FB, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the downlink (send) direction. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims. VirtualSubscriber.Evs.Synch.Select.set.

param send_bw NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.26 Bitrate

class Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.send.bitrate.clone()
```

Subgroups

7.2.1.1.3.27 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:SEND:BITRate:RANGe
```

class Range

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

class RangeStruct

Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Lower end of the range, 5.9 kbit/s to 128 kbit/s
- Bitrate_Higher: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Upper end of the range, 5.9 kbit/s to 128 kbit/s

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:SEND:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.virtualSubscriber.evs.
↪send.bitrate.range.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the downlink (send) direction. The value must be compatible to the selected bandwidth. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.Evs_.Send_.Bitrate_.Range.Range.RangeStruct*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:SEND:BITRate:RANGe
driver.configure.data.control.ims.virtualSubscriber.evs.send.bitrate.range.
↪set(value = [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the downlink (send) direction. The value must be compatible to the selected bandwidth. The setting applies only if the uplink and the downlink are configured separately, see method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.28 Synch

class Synch

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.evs.synch.clone()
```

Subgroups

7.2.1.1.3.29 Select

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:SYNCh:SELect
```

class Select

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.BwRange

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:SYNCh:SELect
value: enums.BwRange = driver.configure.data.control.ims.virtualSubscriber.evs.
↪synch.select.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

(continues on next page)

(continued from previous page)

Selects a configuration mode for the bandwidth and bit-rate settings of the EVS primary mode. The uplink (receive) and the downlink (send) can be configured together (COMMon) or separately (SENDrx) .

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return bw_range: COMMon | SENDrx COMMon The following commands apply: method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.BwCommon.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Common.Bitrate.Range.set SENDrx The following commands apply: method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Send.Bw.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Send.Bitrate.Range.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Receive.Bw.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Receive.Bitrate.Range.set

set(*bw_range: RsCmwDau.enums.BwRange*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:SYNCh:SELect
driver.configure.data.control.ims.virtualSubscriber.evs.synch.select.set(bw_
↪range = enums.BwRange.COMMon, ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

Selects a configuration mode for the bandwidth and bit-rate settings of the EVS primary mode. The uplink (receive) and the downlink (send) can be configured together (COMMon) or separately (SENDrx) .

param bw_range COMMon | SENDrx COMMon The following commands apply: method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.BwCommon.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Common.Bitrate.Range.set SENDrx The following commands apply: method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Send.Bw.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Send.Bitrate.Range.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Receive.Bw.set method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Receive.Bitrate.Range.set

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.30 StartMode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:STARtmode
```

class StartMode

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.StartMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:STARtmode
value: enums.StartMode = driver.configure.data.control.ims.virtualSubscriber.
↪evs.startMode.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the start mode for the EVS codec.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return start_mode: EPRimary | EAMRwbio EVS primary or EVS AMR-WB IO

set(*start_mode: RsCmwDau.enums.StartMode*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:STARtmode
driver.configure.data.control.ims.virtualSubscriber.evs.startMode.set(start_
↪mode = enums.StartMode.EAMRwbio, ims = repcap.Ims.Default, virtualSubscriber␣
↪= repcap.VirtualSubscriber.Default)
```

Selects the start mode for the EVS codec.

param start_mode EPRimary | EAMRwbio EVS primary or EVS AMR-WB IO

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.31 ChawMode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:CHAWmode
```

class ChawMode

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.ChawMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:CHAWmode
value: enums.ChawMode = driver.configure.data.control.ims.virtualSubscriber.evs.
↪chawMode.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'ch-aw-recv' for the EVS codec.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return chaw_mode: DIS | NUSed | TWO | THRee | FIVE | SEVen | NP Disabled, not used, 2, 3, 5, 7, not present

set(*chaw_mode: RsCmwDau.enums.ChawMode*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:CHAWmode
driver.configure.data.control.ims.virtualSubscriber.evs.chawMode.set(chaw_mode␣
↪= enums.ChawMode.DIS, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'ch-aw-recv' for the EVS codec.

param chaw_mode DIS | NUSed | TWO | THRee | FIVE | SEVen | NP Disabled, not used, 2, 3, 5, 7, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.32 Cmr

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:CMR
```

class Cmr

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Cmr

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:EVS:CMR
value: enums.Cmr = driver.configure.data.control.ims.virtualSubscriber.evs.cmr.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Specifies the SDP parameter 'cmr' for the EVS codec.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return cmr: DISable | ENABle | PRESent | NP Disable, enable, present all, not present

set(*cmr: RsCmwDau.enums.Cmr*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:EVS:CMR
driver.configure.data.control.ims.virtualSubscriber.evs.cmr.set(cmr = enums.Cmr.
↪DISable, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'cmr' for the EVS codec.

param cmr DISable | ENABle | PRESent | NP Disable, enable, present all, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.33 DtxRecv

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:DTXRecv
```

class DtxRecv

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.DtxRecv

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:DTXRecv
value: enums.DtxRecv = driver.configure.data.control.ims.virtualSubscriber.evs.
↪dtxRecv.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'dtx-recv' for the EVS codec.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return dtx_recv: DISable | ENABle | NP Disable, enable, not present

set(*dtx_recv: RsCmwDau.enums.DtxRecv*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:DTXRecv
driver.configure.data.control.ims.virtualSubscriber.evs.dtxRecv.set(dtx_recv =␣
↪enums.DtxRecv.DISable, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'dtx-recv' for the EVS codec.

param dtx_recv DISable | ENABle | NP Disable, enable, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.34 Dtx

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:DTX
```

class Dtx

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.DtxRecv

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:EVS:DTX
value: enums.DtxRecv = driver.configure.data.control.ims.virtualSubscriber.evs.
↪dtx.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'dtx' for the EVS codec.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return dtx: DISable | ENABle | NP Disable, enable, not present

set(*dtx: RsCmwDau.enums.DtxRecv*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:EVS:DTX
driver.configure.data.control.ims.virtualSubscriber.evs.dtx.set(dtx = enums.
↪DtxRecv.DISable, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'dtx' for the EVS codec.

param dtx DISable | ENABle | NP Disable, enable, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.35 HfOnly

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:HFONly
```

class HfOnly

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.HfOnly

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:HFONly
value: enums.HfOnly = driver.configure.data.control.ims.virtualSubscriber.evs.
↪hfOnly.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'hf-only' for the EVS codec.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return hf: BOTH | HEADfull | NP Both, header-full only, not present

set(*hf: RsCmwDau.enums.HfOnly*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:HFONly
driver.configure.data.control.ims.virtualSubscriber.evs.hfOnly.set(hf = enums.
↪HfOnly.BOTH, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the SDP parameter 'hf-only' for the EVS codec.

param hf BOTH | HEADfull | NP Both, header-full only, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.36 BwCommon

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:EVS:BWCommon
```

class BwCommon

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:BWCommon
value: enums.Bandwidth = driver.configure.data.control.ims.virtualSubscriber.
↪evs.bwCommon.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the codec bandwidths supported in the EVS primary mode. The setting applies only if the uplink (receive) and the downlink (send) are configured together, see method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return bw_common: NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

set(*bw_common: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:EVS:BWCommon
driver.configure.data.control.ims.virtualSubscriber.evs.bwCommon.set(bw_common␣
↪= enums.Bandwidth.FB, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the codec bandwidths supported in the EVS primary mode. The setting applies only if the uplink (receive) and the downlink (send) are configured together, see method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Evs.Synch.Select.set.

param bw_common NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.37 Conference

class Conference

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

Cloning the Group

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

Subgroups

7.2.1.1.3.38 Factory

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:CONFerence:FACTory
```

class Factory

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:CONFerence:FACTory
value: bool = driver.configure.data.control.ims.virtualSubscriber.conference.
↪factory.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return conference_factory: No help available

set(*conference_factory: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:CONFerence:FACTory
driver.configure.data.control.ims.virtualSubscriber.conference.factory.
↪set(conference_factory = False, ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param conference_factory No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.39 Supported

class Supported

Supported commands group definition. 4 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.40 Features

class Features

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.supported.features.clone()
```

Subgroups

7.2.1.1.3.41 FileTransfer

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:FILetransfer
```

class FileTransfer

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:FILetransfer
value: bool = driver.configure.data.control.ims.virtualSubscriber.supported.
↪features.fileTransfer.get(ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports file transfer via MSRP.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return file_transfer: OFF | ON

set(*file_transfer: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:FILetransfer
driver.configure.data.control.ims.virtualSubscriber.supported.features.
↪fileTransfer.set(file_transfer = False, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports file transfer via MSRP.

param file_transfer OFF | ON

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.42 SessionMode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:SESSionmode
```

class SessionMode

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:SESSionmode
value: bool = driver.configure.data.control.ims.virtualSubscriber.supported.
↪features.sessionMode.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports session mode messaging.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return session_mode: OFF | ON

set(*session_mode: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:SESSionmode
driver.configure.data.control.ims.virtualSubscriber.supported.features.
↪sessionMode.set(session_mode = False, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports session mode messaging.

param session_mode OFF | ON

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.43 Standalone

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:STANdalone
```

class Standalone

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:STANdalone
value: bool = driver.configure.data.control.ims.virtualSubscriber.supported.
↪features.standalone.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports standalone messaging.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return standalone: OFF | ON

set(*standalone: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:STANdalone
driver.configure.data.control.ims.virtualSubscriber.supported.features.
↪standalone.set(standalone = False, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports standalone messaging.

param standalone OFF | ON

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.44 Video

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:SUPPorted:FEATures:VIDeo
```

class Video

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:VIDeo
value: bool = driver.configure.data.control.ims.virtualSubscriber.supported.
↪features.video.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports video calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return video: OFF | ON

set(*video: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SUPPorted:FEATures:VIDeo
driver.configure.data.control.ims.virtualSubscriber.supported.features.video.
↪set(video = False, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies whether the virtual subscriber number <v> supports video calls.

param video OFF | ON

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.45 PcapFile

class PcapFile

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

Cloning the Group

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

Subgroups

7.2.1.1.3.46 Streaming

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:PCAPfile:STReaming
```

class Streaming

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.PcapMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:PCAPfile:STReaming
value: enums.PcapMode = driver.configure.data.control.ims.virtualSubscriber.
↪pcapFile.streaming.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return pcap_mode: No help available

set(*pcap_mode: RsCmwDau.enums.PcapMode*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:PCAPfile:STReaming
driver.configure.data.control.ims.virtualSubscriber.pcapFile.streaming.set(pcap_
↪mode = enums.PcapMode.CYC, ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param pcap_mode No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.47 Selection

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:PCAPfile:SELection
```

class Selection

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:PCAPfile:SELection
value: str = driver.configure.data.control.ims.virtualSubscriber.pcapFile.
↪selection.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects an XML file with metadata for the media endpoint type 'PCAP'. The file must be on the samba share of the DAU, in the subdirectory ims/pcap.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return pcap_file: File name as string

set(*pcap_file: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:PCAPfile:SELection
driver.configure.data.control.ims.virtualSubscriber.pcapFile.selection.set(pcap_
↪file = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects an XML file with metadata for the media endpoint type 'PCAP'. The file must be on the samba share of the DAU, in the subdirectory ims/pcap.

param pcap_file File name as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.48 MtFileTfr

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTFiletfr:SEND
```

class MtFileTfr

MtFileTfr commands group definition. 5 total commands, 4 Sub-groups, 1 group commands

send(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:SEND
driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.send(ims = repcap.
↪Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Initiates a file transfer from virtual subscriber number <v> to the DUT.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

send_with_opc(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

Cloning the Group

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

Subgroups

7.2.1.1.3.49 ChunkSize

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTFiletfr:CHUNksize
```

class ChunkSize

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:CHUNksize
value: int = driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.
↪chunkSize.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the size of the chunks into which the file is divided for transfer.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return chunk_size: Range: 1 byte to 2.147483647E+9 bytes, Unit: byte

set(*chunk_size: int*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:CHUNksize
driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.chunkSize.
↪set(chunk_size = 1, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the size of the chunks into which the file is divided for transfer.

param chunk_size Range: 1 byte to 2.147483647E+9 bytes, Unit: byte

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.50 File

class File

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.file.clone()
```

Subgroups

7.2.1.1.3.51 Selection

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTFiletfr:FILE:SELection
```

class Selection

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:FILE:SELection
value: str = driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.file.
↪selection.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects a file for file transfers by virtual subscriber number <v>. The file must be on the samba share of the DAU, in the subdirectory ims/rcs/samples.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return selected_file: File name as string

set(*selected_file: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:FILE:SELection
driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.file.selection.
↪set(selected_file = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects a file for file transfers by virtual subscriber number <v>. The file must be on the samba share of the DAU, in the subdirectory ims/rcs/samples.

param selected_file File name as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.52 TypePy

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTFiletfr:TYPE
```

class TypePy

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.FileTransferType

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:TYPE
value: enums.FileTransferType = driver.configure.data.control.ims.
↪virtualSubscriber.mtFileTfr.typePy.get(ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the transfer type for file transfers by virtual subscriber number <v>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return file_transfer_type: FILetransfer | LARGe File transfer or file transfer large mode

set(*file_transfer_type: RsCmwDau.enums.FileTransferType*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:TYPE
driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.typePy.set(file_
↪transfer_type = enums.FileTransferType.FILetransfer, ims = repcap.Ims.Default,
↪ virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the transfer type for file transfers by virtual subscriber number <v>.

param file_transfer_type FILetransfer | LARGe File transfer or file transfer large mode

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.53 Destination

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTFiletfr:DESTination
```

class Destination

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:DESTination
value: str = driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.
↪destination.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the destination for file transfers by virtual subscriber number <v>. To query a list of all possible destination strings, see method **RsCmwDau.Sense.Data.Control.Ims.VirtualSubscriber.MtFileTfr.Destination.ListPy.get_**.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return destination: Destination string

set(*destination: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTFiletfr:DESTination
driver.configure.data.control.ims.virtualSubscriber.mtFileTfr.destination.
↪set(destination = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the destination for file transfers by virtual subscriber number <v>. To query a list of all possible destination strings, see method **RsCmwDau.Sense.Data.Control.Ims.VirtualSubscriber.MtFileTfr.Destination.ListPy.get_**.

param destination Destination string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.54 AudioBoard

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:AUDioboard
```

class AudioBoard

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

class AudioBoardStruct

Structure for setting input parameters. Fields:

- Instance: enums.AudioInstance: No parameter help available
- Dtx_Enable: bool: No parameter help available
- Force_Mode_Nb: enums.ForceModeNb: No parameter help available
- Force_Mode_Wb: enums.ForceModeWb: No parameter help available

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → AudioBoardStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AUDioboard
value: AudioBoardStruct = driver.configure.data.control.ims.virtualSubscriber.
↪audioBoard.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.AudioBoard.AudioBoard.AudioBoardStruct*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AUDioboard
driver.configure.data.control.ims.virtualSubscriber.audioBoard.set(value =␣
↪[PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

Cloning the Group

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

Subgroups

7.2.1.1.3.55 Config

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:AUDioboard:CONFig
```

class Config

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

class ConfigStruct

Structure for setting input parameters. Fields:

- Instance: enums.AudioInstance: INST1 | INST2 Audio software instance 1 or 2
- Dtx_Enable: bool: OFF | ON Enable comfort noise in the downlink for AMR codecs
- Force_Mode_Nb: enums.ForceModeNb: ZERO | ONE | TWO | THRE | FOUR | FIVE | SIX | SEVN | FREE Index of the codec rate to be used if the AMR narrowband codec is active FREE means that no specific codec rate is forced
- Force_Mode_Wb: enums.ForceModeWb: ZERO | ONE | TWO | THRE | FOUR | FIVE | SIX | SEVN | EIGH | FREE Index of the codec rate to be used if the AMR wideband codec is active FREE means that no specific codec rate is forced

- Force_Mode_Evs: enums.ForceModeEvs: SDP | P28 | P72 | P80 | P96 | P132 | P164 | P244 | P320 | P480 | P640 | P960 | P1280 | A660 | A885 | A1265 | A1425 | A1585 | A1825 | A1985 | A2305 | A2385 Start mode and rate to be used if the EVS codec is active SDP: no specific codec rate forced P28 to P1280: EVS primary mode, 2.8 kbit/s to 128 kbit/s A660 to A2385: AMR-WB IO mode, 6.6 kbit/s to 23.85 kbit/s

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → ConfigStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AUDioboard:CONFig
value: ConfigStruct = driver.configure.data.control.ims.virtualSubscriber.
↪audioBoard.config.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Configures the audio board. INTRO_CMD_HELP: A query returns only the values that are relevant for the active codec:

- NB AMR: <Instance>, <DTXEnable>, <ForceModeNB>
- WB AMR: <Instance>, <DTXEnable>, <ForceModeWB>
- EVS: <Instance>, <ForceModeEVS>

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.AudioBoard_.Config.Config.ConfigStruct*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AUDioboard:CONFig
driver.configure.data.control.ims.virtualSubscriber.audioBoard.config.set(value␣
↪= [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

Configures the audio board. INTRO_CMD_HELP: A query returns only the values that are relevant for the active codec:

- NB AMR: <Instance>, <DTXEnable>, <ForceModeNB>
- WB AMR: <Instance>, <DTXEnable>, <ForceModeWB>
- EVS: <Instance>, <ForceModeEVS>

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.56 ForceMoCall

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:FORCemocall
```

class ForceMoCall

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:FORCemocall
value: bool = driver.configure.data.control.ims.virtualSubscriber.forceMoCall.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Forces a specific codec type for mobile-originating calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return force_codec_mo: OFF | ON OFF The first codec type offered via SDP is used. ON The voice codec type configured for the virtual subscriber is used.

set(*force_codec_mo: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:FORCemocall
driver.configure.data.control.ims.virtualSubscriber.forceMoCall.set(force_codec_
↪mo = False, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Forces a specific codec type for mobile-originating calls.

param force_codec_mo OFF | ON OFF The first codec type offered via SDP is used. ON The voice codec type configured for the virtual subscriber is used.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.57 Bearer

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:BEARer
```

class Bearer

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:BEARer
value: bool = driver.configure.data.control.ims.virtualSubscriber.bearer.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Enables the automatic setup of a dedicated bearer.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return bearer: OFF | ON

set(*bearer: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:BEARer
driver.configure.data.control.ims.virtualSubscriber.bearer.set(bearer = False,␣
↪ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Enables the automatic setup of a dedicated bearer.

param bearer OFF | ON

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.58 Id

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:ID
```

class Id

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:ID
value: str = driver.configure.data.control.ims.virtualSubscriber.id.get(ims =␣
↪repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Specifies the public user ID of the virtual subscriber number <v>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return idn: Public user ID as string

set(*idn: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:ID
driver.configure.data.control.ims.virtualSubscriber.id.set(idn = '1', ims =␣
↪repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Specifies the public user ID of the virtual subscriber number <v>.

param idn Public user ID as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.59 Behaviour

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:BEHaviour
```

class Behaviour

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.BehaviourA

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:BEHaviour
value: enums.BehaviourA = driver.configure.data.control.ims.virtualSubscriber.
↪behaviour.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Defines the reaction of virtual subscriber number <v> to incoming calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return behaviour: ANSWer | NOANswer | DECLined | BUSY | BEFRng | AFTRng | CD ANSWer: answer the call NOANswer: keep 'ringing' DECLined: reject call BUSY: subscriber busy BEFRng: call forwarding before ringing AFTRng: call forwarding after ringing CD: communication deflection

set(*behaviour: RsCmwDau.enums.BehaviourA*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:BEHaviour
driver.configure.data.control.ims.virtualSubscriber.behaviour.set(behaviour =␣
↪enums.BehaviourA.AFTRng, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Defines the reaction of virtual subscriber number <v> to incoming calls.

param behaviour ANSWer | NOANswer | DECLined | BUSY | BEFRng | AFTRng | CD ANSWer: answer the call NOANswer: keep 'ringing' DECLined: reject call BUSY: subscriber busy BEFRng: call forwarding before ringing AFTRng: call forwarding after ringing CD: communication deflection

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.60 SignalingType

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:SIGNalingtyp
```

class SignalingType

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.SignalingType

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SIGNalingtyp
value: enums.SignalingType = driver.configure.data.control.ims.
↪virtualSubscriber.signalingType.get(ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Specifies whether a voice call session is established with or without quality-of-service preconditions.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return sig_type: PRECondit | NOPRecondit | SIMPle | REQU100 | REQuprecondi | WOTPrec183 | EARLymedia PRECondit With preconditions NOPRecondit Without

preconditions SIMPle Simplified call flow, without preconditions REQU100 Require 100rel, without preconditions REQuprecondi Require precondition, with preconditions WOTPrec183 With 183, without preconditions EARLymedia Early media, with or without preconditions

set(*sig_type: RsCmwDau.enums.SignalingType*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:SIGNalingtyp
driver.configure.data.control.ims.virtualSubscriber.signalingType.set(sig_type␣
↪= enums.SignalingType.EARLymedia, ims = repcap.Ims.Default, virtualSubscriber␣
↪= repcap.VirtualSubscriber.Default)
```

Specifies whether a voice call session is established with or without quality-of-service preconditions.

param sig_type PRECondit | NOPRecondit | SIMPle | REQU100 | REQuprecondi | WOTPrec183 | EARLymedia PRECondit With preconditions NOPRecondit Without preconditions SIMPle Simplified call flow, without preconditions REQU100 Require 100rel, without preconditions REQuprecondi Require precondition, with preconditions WOTPrec183 With 183, without preconditions EARLymedia Early media, with or without preconditions

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.61 AdCodec

class AdCodec

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

Cloning the Group

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

Subgroups

7.2.1.1.3.62 TypePy

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:ADCodec:TYPE
```

class TypePy

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.CodecType

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:ADCodec:TYPE
value: enums.CodecType = driver.configure.data.control.ims.virtualSubscriber.
↪adCodec.typePy.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the audio codec type for voice over IMS calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return type_py: NARRowband | WIDeband | EVS AMR NB, AMR WB, EVS

set(*type_py: RsCmwDau.enums.CodecType*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:ADCodec:TYPE
driver.configure.data.control.ims.virtualSubscriber.adCodec.typePy.set(type_py␣
↪= enums.CodecType.EVS, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the audio codec type for voice over IMS calls.

param type_py NARRowband | WIDeband | EVS AMR NB, AMR WB, EVS

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.63 Amr

class Amr

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

Cloning the Group

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

Subgroups

7.2.1.1.3.64 Alignment

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:AMR:ALIGnment
```

class Alignment

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.AlignMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AMR:ALIGnment
value: enums.AlignMode = driver.configure.data.control.ims.virtualSubscriber.
↪amr.alignment.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the AMR voice codec alignment mode for voice over IMS calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return align_mode: OCTetaligned | BANDwidtheff OCTetaligned: octet-aligned BANDwidtheff: bandwidth-efficient

set(*align_mode: RsCmwDau.enums.AlignMode*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AMR:ALIGnment
driver.configure.data.control.ims.virtualSubscriber.amr.alignment.set(align_
↪mode = enums.AlignMode.BANDwidtheff, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the AMR voice codec alignment mode for voice over IMS calls.

param align_mode OCTetaligned | BANDwidtheff OCTetaligned: octet-aligned BANDwidtheff: bandwidth-efficient

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.65 Codec<Codec>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.virtualSubscriber.amr.codec.repcap_codec_get()
driver.configure.data.control.ims.virtualSubscriber.amr.codec.repcap_codec_set(repcap.
↪Codec.Ix1)
```

class Codec

Codec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Codec, default value after init: Codec.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.amr.codec.clone()
```

Subgroups

7.2.1.1.3.66 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:AMR:CODec<Codec>:ENABle
```

class Enable

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AMR:CODec<CodecIdx>:ENABle
value: bool = driver.configure.data.control.ims.virtualSubscriber.amr.codec.
↪enable.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

Enables or disables a codec rate for the currently active AMR type, see method RsCmwDau.Configure.Data.Control.Ims. VirtualSubscriber.AdCodec.TypePy.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

return codec_rate: OFF | ON OFF: codec rate not supported ON: codec rate supported

set(*codec_rate: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:AMR:CODec<CodecIdx>:ENABle
driver.configure.data.control.ims.virtualSubscriber.amr.codec.enable.set(codec_
↪rate = False, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

Enables or disables a codec rate for the currently active AMR type, see method RsCmwDau.Configure.Data.Control.Ims. VirtualSubscriber.AdCodec.TypePy.set.

param codec_rate OFF | ON OFF: codec rate not supported ON: codec rate supported

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

7.2.1.1.3.67 Video

class Video

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

Cloning the Group

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

Subgroups

7.2.1.1.3.68 Codec

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:VIDeo:CODec
```

class Codec

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.VideoCodec

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:VIDeo:CODec
value: enums.VideoCodec = driver.configure.data.control.ims.virtualSubscriber.
↪video.codec.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

(continues on next page)

(continued from previous page)

Selects the codec for video calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return codec: H263 | H264 H.263 or H.264 codec

set(*codec: RsCmwDau.enums.VideoCodec*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:VIDeo:CODec
driver.configure.data.control.ims.virtualSubscriber.video.codec.set(codec =␣
↪enums.VideoCodec.H263, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the codec for video calls.

param codec H263 | H264 H.263 or H.264 codec

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.69 Attributes

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:VIDeo:ATTRibutes
```

class Attributes

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:VIDeo:ATTRibutes
value: str = driver.configure.data.control.ims.virtualSubscriber.video.
↪attributes.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Configures codec attributes for video calls.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return attributes: Codec attributes as string

set(*attributes: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:VIDeo:ATTRibutes
driver.configure.data.control.ims.virtualSubscriber.video.attributes.
↪set(attributes = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Configures codec attributes for video calls.

param attributes Codec attributes as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.70 MediaEndpoint

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MEDiaendpoin
```

class MediaEndpoint

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.MediaEndpoint

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MEDiaendpoin
value: enums.MediaEndpoint = driver.configure.data.control.ims.
↪virtualSubscriber.mediaEndpoint.get(ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Configures the media endpoint.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return media_endpoint: LOOPback | FORWard | AUDioboard | PCAP LOOPback: Loop back to the DUT FORWard: Route to an external media endpoint AUDioboard: Route to the speech codec of the audio board PCAP: Play a PCAP file

set(*media_endpoint: RsCmwDau.enums.MediaEndpoint*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MEDiaendpoin
driver.configure.data.control.ims.virtualSubscriber.mediaEndpoint.set(media_
↪endpoint = enums.MediaEndpoint.AUDioboard, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Configures the media endpoint.

param media_endpoint LOOPback | FORWard | AUDioboard | PCAP LOOPback: Loop back to the DUT FORWard: Route to an external media endpoint AUDioboard: Route to the speech codec of the audio board PCAP: Play a PCAP file

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.71 Forward

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:FORWard
```

class Forward

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

class ForwardStruct

Structure for setting input parameters. Fields:

- Ip_Address: str: IPv4 or IPv6 address of the media endpoint as string
- Port: int: Port for RTP packet forwarding
- Cmd_Port: int: Port for the command interface to the media endpoint
- Amr_Align: enums.AlignMode: OCTetaligned | BANDwidtheff AMR alignment mode used by the media endpoint OCTetaligned: octet-aligned BANDwidtheff: bandwidth-efficient

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → ForwardStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:FORWard
value: ForwardStruct = driver.configure.data.control.ims.virtualSubscriber.
↪forward.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Configures an external media endpoint.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.Forward.Forward.ForwardStruct*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>:FORWard
driver.configure.data.control.ims.virtualSubscriber.forward.set(value =␣
↪[PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

Configures an external media endpoint.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.72 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub:ADD
```

class Add

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

set(*ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub:ADD
driver.configure.data.control.ims.virtualSubscriber.add.set(ims = repcap.Ims.
↪Default)
```

Creates a new virtual subscriber profile. See also method RsCmwDau.Configure.Data.Control.Ims.VirtualSubscriber.Create. set

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims=<Ims.Default: -1>*) → None

7.2.1.1.3.73 Create

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub:CREate
```

class Create

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

set(*ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub:CREate
driver.configure.data.control.ims.virtualSubscriber.create.set(ims = repcap.Ims.
↪Default)
```

Updates the internal list of virtual subscriber profiles. If your command script adds virtual subscriber profiles, you must insert this command before you can use the new profiles. It is sufficient to insert the command once, after adding the last virtual subscriber profile / before using the profiles.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims=<Ims.Default: -1>*) → None

7.2.1.1.3.74 MtSms

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTSMs:SEND
```

class MtSms

MtSms commands group definition. 6 total commands, 5 Sub-groups, 1 group commands

send(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:SEND
driver.configure.data.control.ims.virtualSubscriber.mtSms.send(ims = repcap.Ims.
↪Default, virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Initiates a message transfer from virtual subscriber number <v> to the DUT.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

send_with_opc(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

Cloning the Group

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

Subgroups

7.2.1.1.3.75 ImportPy

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtSms.importPy.clone()
```

Subgroups

7.2.1.1.3.76 File

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTSMs:IMPort:FILE
```

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:IMPort:FILE
value: str = driver.configure.data.control.ims.virtualSubscriber.mtSms.importPy.
↪file.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Imports the message text for 'RCS Large Mode' transfer from a file, for virtual subscriber number <v>. The file must be on the samba share of the DAU, in the subdirectory ims/rcs/samples.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return selected_file: File name as string

set(*selected_file: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:IMPort:FILE
driver.configure.data.control.ims.virtualSubscriber.mtSms.importPy.file.
↪set(selected_file = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Imports the message text for 'RCS Large Mode' transfer from a file, for virtual subscriber number <v>. The file must be on the samba share of the DAU, in the subdirectory ims/rcs/samples.

param selected_file File name as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.77 Encoding

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTSMs:ENCoding
```

class Encoding

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.MtSmsEncoding

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:ENCoding
value: enums.MtSmsEncoding = driver.configure.data.control.ims.
↪virtualSubscriber.mtSms.encoding.get(ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Selects the encoding for RCS messages.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return encoding: NENCoding | BASE64 No encoding or Base64 encoding

set(*encoding: RsCmwDau.enums.MtSmsEncoding*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:ENCoding
driver.configure.data.control.ims.virtualSubscriber.mtSms.encoding.set(encoding␣
↪= enums.MtSmsEncoding.BASE64, ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

Selects the encoding for RCS messages.

param encoding NENCoding | BASE64 No encoding or Base64 encoding

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.78 Destination

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTSMs:DESTination
```

class Destination

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:DESTination
value: str = driver.configure.data.control.ims.virtualSubscriber.mtSms.
↪destination.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the destination to which messages are sent by virtual subscriber number <v>. To query a list of all possible destination strings, see method **RsCmwDau.Sense.Data.Control.Ims.VirtualSubscriber.MtSms.Destination.ListPy.get_**.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return destination: Destination string

set(*destination: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:DESTination
driver.configure.data.control.ims.virtualSubscriber.mtSms.destination.
↪set(destination = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the destination to which messages are sent by virtual subscriber number <v>. To query a list of all possible destination strings, see method **RsCmwDau.Sense.Data.Control.Ims.VirtualSubscriber.MtSms.Destination.ListPy.get_**.

param destination Destination string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.79 TypePy

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTSMs:TYPE
```

class TypePy

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.CallType

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:TYPE
value: enums.CallType = driver.configure.data.control.ims.virtualSubscriber.
↪mtSms.typePy.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the type of messages to be sent by virtual subscriber number <v>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return type_py: GPP | GPP2 | ACK | PAGer | LARGe | RCSChat | RCSGrpchat | GENeric GPP: 3GPP GPP2: 3GPP2 without delivery ACK ACK: 3GPP2 with delivery ACK PAGer: RCS pager mode LARGe: RCS large mode RCSChat: RCS 1 to 1 chat RCSGrpchat: RCS group chat GENeric: 3GPP generic SMS

set(*type_py: RsCmwDau.enums.CallType*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:TYPE
driver.configure.data.control.ims.virtualSubscriber.mtSms.typePy.set(type_py =␣
↪enums.CallType.ACK, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the type of messages to be sent by virtual subscriber number <v>.

param type_py GPP | GPP2 | ACK | PAGer | LARGe | RCSChat | RCSGrpchat | GENeric GPP: 3GPP GPP2: 3GPP2 without delivery ACK ACK: 3GPP2 with delivery ACK PAGer: RCS pager mode LARGe: RCS large mode RCSChat: RCS 1 to 1 chat RCSGrpchat: RCS group chat GENeric: 3GPP generic SMS

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.80 Text

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTSMs:TEXT
```

class Text

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:TEXT
value: str = driver.configure.data.control.ims.virtualSubscriber.mtSms.text.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Defines the text for messages to be sent by virtual subscriber number <v>. For generic SMS, it defines the contents of the RPDU via hexadecimal characters.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return text: Message text or RPDU contents as string

set(*text: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTSMs:TEXT
driver.configure.data.control.ims.virtualSubscriber.mtSms.text.set(text = '1',␣
↪ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

Defines the text for messages to be sent by virtual subscriber number <v>. For generic SMS, it defines the contents of the RPDU via hexadecimal characters.

param text Message text or RPDU contents as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.81 MtCall

class MtCall

MtCall commands group definition. 25 total commands, 10 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.82 Sdp

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:SDP
```

class Sdp

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

class SdpStruct

Structure for setting input parameters. Fields:

- Sdp_Enable: bool: No parameter help available
- Sdp_File_Name: str: No parameter help available

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → SdpStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:SDP
value: SdpStruct = driver.configure.data.control.ims.virtualSubscriber.mtCall.
↪sdp.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.MtCall_.Sdp.Sdp.SdpStruct*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:SDP
driver.configure.data.control.ims.virtualSubscriber.mtCall.sdp.set(value =␣
↪[PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.83 Evs

class Evs

Evs commands group definition. 14 total commands, 12 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.clone()
```

Subgroups

7.2.1.1.3.84 Codec<Codec>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.codec.repcap_codec_
↪get()
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.codec.repcap_codec_
↪set(repcap.Codec.Ix1)
```

class Codec

Codec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Codec, default value after init: Codec.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.codec.clone()
```

Subgroups

7.2.1.1.3.85 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:CODec<Codec>
↪:ENABle
```

class Enable

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:CODec<CodecIdx>:ENABle
value: bool = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.
↪codec.enable.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

return codec_rate: No help available

set(*codec_rate: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:CODec<CodecIdx>:ENABle
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.codec.enable.
↪set(codec_rate = False, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

No command help available

param codec_rate No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

7.2.1.1.3.86 Common

class Common

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.common.clone()
```

Subgroups

7.2.1.1.3.87 Bitrate

class Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.common.bitrate.
↪clone()
```

Subgroups

7.2.1.1.3.88 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:COMMon:BITRate:RANGe
```

class Range

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

class RangeStruct

Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: No parameter help available
- Bitrate_Higher: enums.Bitrate: No parameter help available

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:COMMon:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.virtualSubscriber.mtCall.
↪evs.common.bitrate.range.get(ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.MtCall_.Evs_.Common_.Bitrate_.Range.Range.I*
ims=<Ims.Default: -1>, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:COMMon:BITRate:RANGe
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.common.bitrate.
↪range.set(value = [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.89 Receive

class Receive

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.receive.clone()
```

Subgroups

7.2.1.1.3.90 Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.receive.bitrate.
↪clone()
```

Subgroups

7.2.1.1.3.91 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:RECeive:BITRate:RANGe
```

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

class RangeStruct
Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: No parameter help available
- Bitrate_Higher: enums.Bitrate: No parameter help available

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:RECeive:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.virtualSubscriber.mtCall.
↪evs.receive.bitrate.range.get(ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.MtCall_.Evs_.Receive_.Bitrate_.Range.Range.R*
ims=<Ims.Default: -1>, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:RECeive:BITRate:RANGe
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.receive.bitrate.
↪range.set(value = [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.92 Bw

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:RECeive:BW
```

class Bw

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:RECeive:BW
value: enums.Bandwidth = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.receive.bw.get(ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return rx_bw: No help available

set(*rx_bw: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:RECeive:BW
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.receive.bw.
↪set(rx_bw = enums.Bandwidth.FB, ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param rx_bw No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.93 Send

class Send

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.send.clone()
```

Subgroups

7.2.1.1.3.94 Bw

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:SEND:BW
```

class Bw

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:SEND:BW
value: enums.Bandwidth = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.send.bw.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return send_bw: No help available

set(*send_bw: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:SEND:BW
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.send.bw.set(send_
↪bw = enums.Bandwidth.FB, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param send_bw No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.95 Bitrate

class Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.send.bitrate.
↪clone()
```

Subgroups

7.2.1.1.3.96 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:SEND:BITRate:RANGe
```

class Range

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

class RangeStruct

Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: No parameter help available
- Bitrate_Higher: enums.Bitrate: No parameter help available

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:SEND:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.virtualSubscriber.mtCall.
↪evs.send.bitrate.range.get(ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

(continues on next page)

(continued from previous page)

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.VirtualSubscriber_.MtCall_.Evs_.Send_.Bitrate_.Range.Range.Rang*
ims=<Ims.Default: -1>, virtualSubscriber=<VirtualSubscriber.Default: -1>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:SEND:BITRate:RANGe
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.send.bitrate.
↪range.set(value = [PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.97 Synch

class Synch

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.synch.clone()
```

Subgroups

7.2.1.1.3.98 Select

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:SYNCh:SELect
```

class Select

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.BwRange

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:SYNCh:SELect
value: enums.BwRange = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.synch.select.get(ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return bw_ranges: No help available

set(*bw_ranges: RsCmwDau.enums.BwRange*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:SYNCh:SELect
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.synch.select.
↪set(bw_ranges = enums.BwRange.COMMon, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param bw_ranges No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.99 StartMode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:STARtmode
```

class StartMode

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.StartMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:STARtmode
value: enums.StartMode = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.startMode.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

(continues on next page)

(continued from previous page)

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return start_mode: No help available

set(*start_mode: RsCmwDau.enums.StartMode*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:STARtmode
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.startMode.
↪set(start_mode = enums.StartMode.EAMRwbio, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param start_mode No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.100 ChawMode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:CHAWmode
```

class ChawMode

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.ChawMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:CHAWmode
value: enums.ChawMode = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.chawMode.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return chaw_mode: No help available

set(*chaw_mode: RsCmwDau.enums.ChawMode*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:CHAWmode
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.chawMode.
↪set(chaw_mode = enums.ChawMode.DIS, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param chaw_mode No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.101 Cmr

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:CMR
```

class Cmr

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Cmr

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:CMR
value: enums.Cmr = driver.configure.data.control.ims.virtualSubscriber.mtCall.
↪evs.cmr.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return cmr: No help available

set(*cmr: RsCmwDau.enums.Cmr*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:CMR
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.cmr.set(cmr =␣
↪enums.Cmr.DISable, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param cmr No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.102 DtxRecv

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:DTXRecv
```

class DtxRecv

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.DtxRecv

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:DTXRecv
value: enums.DtxRecv = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.dtxRecv.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return dtx_recv: No help available

set(*dtx_recv: RsCmwDau.enums.DtxRecv*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:DTXRecv
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.dtxRecv.set(dtx_
↪recv = enums.DtxRecv.DISable, ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param dtx_recv No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.103 Dtx

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:DTX
```

class Dtx

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.DtxRecv

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:DTX
value: enums.DtxRecv = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.dtx.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return dtx: No help available

set(*dtx: RsCmwDau.enums.DtxRecv*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:DTX
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.dtx.set(dtx =␣
↪enums.DtxRecv.DISable, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param dtx No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.104 HfOnly

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:HFONly
```

class HfOnly

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.HfOnly

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:HFONly
value: enums.HfOnly = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.hfOnly.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return hf: No help available

set(*hf: RsCmwDau.enums.HfOnly*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:HFONly
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.hfOnly.set(hf =␣
↪enums.HfOnly.BOTH, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param hf No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.105 BwCommon

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:EVS:BWCommon
```

class BwCommon

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:BWCommon
value: enums.Bandwidth = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.evs.bwCommon.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return bw_common: No help available

set(*bw_common: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:EVS:BWCommon
driver.configure.data.control.ims.virtualSubscriber.mtCall.evs.bwCommon.set(bw_
↪common = enums.Bandwidth.FB, ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param bw_common No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.106 Bearer

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:BEARer
```

class Bearer

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:BEARer
value: bool = driver.configure.data.control.ims.virtualSubscriber.mtCall.bearer.
↪get(ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return bearer: No help available

set(*bearer: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:BEARer
driver.configure.data.control.ims.virtualSubscriber.mtCall.bearer.set(bearer =␣
↪False, ims = repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.
↪Default)
```

No command help available

param bearer No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.107 Destination

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:DESTination
```

class Destination

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:DESTination
value: str = driver.configure.data.control.ims.virtualSubscriber.mtCall.
↪destination.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the destination to be called by virtual subscriber number <v>. To query a list of all possible destination strings, see method **RsCmwDau.Sense.Data.Control.Ims.VirtualSubscriber.MtCall.Destination.ListPy.get_**.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return destination: Destination string

set(*destination: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:DESTination
driver.configure.data.control.ims.virtualSubscriber.mtCall.destination.
↪set(destination = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Specifies the destination to be called by virtual subscriber number <v>. To query a list of all possible destination strings, see method **RsCmwDau.Sense.Data.Control.Ims.VirtualSubscriber.MtCall.Destination.ListPy.get_**.

param destination Destination string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.108 TypePy

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:TYPE
```

class TypePy

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.AvTypeA

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:TYPE
value: enums.AvTypeA = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.typePy.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the type of call to be initiated: audio call or video call.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return type_py: AUDio | VIDeo

set(*type_py: RsCmwDau.enums.AvTypeA*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:TYPE
driver.configure.data.control.ims.virtualSubscriber.mtCall.typePy.set(type_py =␣
↪enums.AvTypeA.AUDio, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

Selects the type of call to be initiated: audio call or video call.

param type_py AUDio | VIDeo

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.109 SignalingType

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:SIGType
```

class SignalingType

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.SignalingType

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:SIGType
value: enums.SignalingType = driver.configure.data.control.ims.
↪virtualSubscriber.mtCall.signalingType.get(ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return sig_type: No help available

set(*sig_type: RsCmwDau.enums.SignalingType*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:SIGType
driver.configure.data.control.ims.virtualSubscriber.mtCall.signalingType.
↪set(sig_type = enums.SignalingType.EARLymedia, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param sig_type No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.110 AdCodec

class AdCodec

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.adCodec.clone()
```

Subgroups

7.2.1.1.3.111 TypePy

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:ADCodec:TYPE
```

class TypePy

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.CodecType

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:ADCodec:TYPE
value: enums.CodecType = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.adCodec.typePy.get(ims = repcap.Ims.Default, virtualSubscriber =␣
↪repcap.VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return type_py: No help available

set(*type_py: RsCmwDau.enums.CodecType*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:ADCodec:TYPE
driver.configure.data.control.ims.virtualSubscriber.mtCall.adCodec.typePy.
↪set(type_py = enums.CodecType.EVS, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param type_py No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.112 Amr

class Amr

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.amr.clone()
```

Subgroups

7.2.1.1.3.113 Alignment

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:AMR:ALIGnment
```

class Alignment

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.AlignMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:AMR:ALIGnment
value: enums.AlignMode = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.amr.alignment.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return alignment_mode: No help available

set(*alignment_mode: RsCmwDau.enums.AlignMode*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:AMR:ALIGnment
driver.configure.data.control.ims.virtualSubscriber.mtCall.amr.alignment.
↪set(alignment_mode = enums.AlignMode.BANDwidtheff, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param alignment_mode No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.114 Codec<Codec>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.virtualSubscriber.mtCall.amr.codec.repcap_codec_
↪get()
driver.configure.data.control.ims.virtualSubscriber.mtCall.amr.codec.repcap_codec_
↪set(repcap.Codec.Ix1)
```

class Codec

Codec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Codec, default value after init: Codec.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.amr.codec.clone()
```

Subgroups

7.2.1.1.3.115 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:AMR:CODec<Codec>
↪:ENABle
```

class Enable

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:AMR:CODec<CodecIdx>:ENABle
value: bool = driver.configure.data.control.ims.virtualSubscriber.mtCall.amr.
↪codec.enable.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

return codec_rate: No help available

set(*codec_rate: bool*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*, *codec=<Codec.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:AMR:CODec<CodecIdx>:ENABle
driver.configure.data.control.ims.virtualSubscriber.mtCall.amr.codec.enable.
↪set(codec_rate = False, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default, codec = repcap.Codec.Default)
```

No command help available

param codec_rate No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

7.2.1.1.3.116 Video

class Video

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.virtualSubscriber.mtCall.video.clone()
```

Subgroups

7.2.1.1.3.117 Codec

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:VIDeo:CODec
```

class Codec

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → RsCmwDau.enums.VideoCodec

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:VIDeo:CODec
value: enums.VideoCodec = driver.configure.data.control.ims.virtualSubscriber.
↪mtCall.video.codec.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return codec: No help available

set(*codec: RsCmwDau.enums.VideoCodec*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:VIDeo:CODec
driver.configure.data.control.ims.virtualSubscriber.mtCall.video.codec.
↪set(codec = enums.VideoCodec.H263, ims = repcap.Ims.Default,␣
↪virtualSubscriber = repcap.VirtualSubscriber.Default)
```

No command help available

param codec No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.118 Attributes

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:VIDeo:ATTRibutes
```

class Attributes

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:VIDeo:ATTRibutes
value: str = driver.configure.data.control.ims.virtualSubscriber.mtCall.video.
↪attributes.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return attributes: No help available

set(*attributes: str*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:VIDeo:ATTRibutes
driver.configure.data.control.ims.virtualSubscriber.mtCall.video.attributes.
↪set(attributes = '1', ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param attributes No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.119 Call

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MTCall:CALL
```

class Call

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

set(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MTCall:CALL
driver.configure.data.control.ims.virtualSubscriber.mtCall.call.set(ims =␣
↪repcap.Ims.Default, virtualSubscriber = repcap.VirtualSubscriber.Default)
```

Initiates the setup of a voice over IMS call, from virtual subscriber number <v> to the DUT.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

set_with_opc(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

7.2.1.1.3.120 Max

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

Cloning the Group

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

Subgroups

7.2.1.1.3.121 Participant

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:VIRTualsub<VirtualSubscriber>:MAX:PARTicipant
```

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

get(*ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MAX:PARTicipant
value: int = driver.configure.data.control.ims.virtualSubscriber.max.
↪participant.get(ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return no_particapant: No help available

set(*no_particapant: int*, *ims=<Ims.Default: -1>*, *virtualSubscriber=<VirtualSubscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:VIRTualsub<VirtualSubscriber>
↪:MAX:PARTicipant
driver.configure.data.control.ims.virtualSubscriber.max.participant.set(no_
↪particapant = 1, ims = repcap.Ims.Default, virtualSubscriber = repcap.
↪VirtualSubscriber.Default)
```

No command help available

param no_particapant No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.122 Sip

class Sip

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

Cloning the Group

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

Subgroups

7.2.1.1.3.123 Timer

class Timer

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

Cloning the Group

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

Subgroups

7.2.1.1.3.124 Case

class Case

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.sip.timer.case.clone()
```

Subgroups

7.2.1.1.3.125 Selection

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SIP:TIMer:CASE:SELection
```

class Selection

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.SipTimerSel

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SIP:TIMer:CASE:SELection
value: enums.SipTimerSel = driver.configure.data.control.ims.sip.timer.case.
↪selection.get(ims = repcap.Ims.Default)
```

Selects a configuration mode for SIP timer T1.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return sip_timer_sel: DEFault | RFC | CUSTom DEFault: 3GPP TS 24.229 (2000 ms) RFC: RFC 3261 (500 ms) CUSTom: method RsCmwDau.Configure.Data.Control.Ims.Sip.Timer.Value.set

set(*sip_timer_sel: RsCmwDau.enums.SipTimerSel*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SIP:TIMer:CASE:SELection
driver.configure.data.control.ims.sip.timer.case.selection.set(sip_timer_sel =␣
↪enums.SipTimerSel.CUSTom, ims = repcap.Ims.Default)
```

Selects a configuration mode for SIP timer T1.

param sip_timer_sel DEFault | RFC | CUSTom DEFault: 3GPP TS 24.229 (2000 ms) RFC: RFC 3261 (500 ms) CUSTom: method RsCmwDau.Configure.Data.Control.Ims.Sip.Timer.Value.set

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.126 Value

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SIP:TIMer:VALue
```

class Value

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

get(*ims=<Ims.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SIP:TIMer:VALue
value: int = driver.configure.data.control.ims.sip.timer.value.get(ims = repcap.
↪Ims.Default)
```

Sets SIP timer T1 for custom mode, see method RsCmwDau.Configure.Data.Control.Ims.Sip.Timer.Case.Selection.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return sip_timer_value: Range: 10 ms to 600E+3 ms, Unit: ms

set(*sip_timer_value: int*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SIP:TIMer:VALue
driver.configure.data.control.ims.sip.timer.value.set(sip_timer_value = 1, ims␣
↪= repcap.Ims.Default)
```

Sets SIP timer T1 for custom mode, see method RsCmwDau.Configure.Data.Control.Ims.Sip.Timer.Case.Selection.set.

param sip_timer_value Range: 10 ms to 600E+3 ms, Unit: ms

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.127 Rcs

class Rcs

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

Cloning the Group

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

Subgroups

7.2.1.1.3.128 GrpChat

class GrpChat

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

Cloning the Group

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

Subgroups

7.2.1.1.3.129 Participant

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:RCS:GRPChat:PARTicipant:DELete
```

class Participant

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

delete(*participant: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:RCS:GRPChat:PARTicipant:DELete
driver.configure.data.control.ims.rcs.grpChat.participant.delete(participant =
↪'1', ims = repcap.Ims.Default)
```

Removes an entry from the list of participants for group chats.

param participant String to be removed

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.rcs.grpChat.participant.clone()
```

Subgroups

7.2.1.1.3.130 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:RCS:GRPChat:PARTicipant:ADD
```

class Add

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

set(*participant: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:RCS:GRPChat:PARTicipant:ADD
driver.configure.data.control.ims.rcs.grpChat.participant.add.set(participant =
↪'1', ims = repcap.Ims.Default)
```

Adds an entry to the list of participants for group chats.

param participant String to be added

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.131 Conference

class Conference

Conference commands group definition. 3 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.132 Factory

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:CONFerence:FACTory:DELete
```

class Factory

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

delete(*factory: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:CONFerence:FACTory:DELete
driver.configure.data.control.ims.conference.factory.delete(factory = '1', ims␣
↪= repcap.Ims.Default)
```

Deletes an entry of the factory list (list of reachable conference server addresses) .

param factory Conference server address to be deleted, as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

Cloning the Group

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

Subgroups

7.2.1.1.3.133 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:CONFerence:FACTory:ADD
```

class Add

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

set(*factory: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:CONFerence:FACTory:ADD
driver.configure.data.control.ims.conference.factory.add.set(factory = '1', ims␣
↪= repcap.Ims.Default)
```

Adds an entry to the factory list (list of reachable conference server addresses) .

param factory Conference server address to be added, as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.134 Max

class Max

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

Cloning the Group

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

Subgroups

7.2.1.1.3.135 Participant

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:CONFerence:MAX:PARTicipant
```

class Participant

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

get(*ims=<Ims.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:CONFerence:MAX:PARTicipant
value: int = driver.configure.data.control.ims.conference.max.participant.
↪get(ims = repcap.Ims.Default)
```

Configures the maximum number of virtual subscribers allowed to participate in a conference call.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return max_participant: The value 0 means that there is no limit. Range: 0 to 10

set(*max_participant: int*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:CONFerence:MAX:PARTicipant
driver.configure.data.control.ims.conference.max.participant.set(max_
↪participant = 1, ims = repcap.Ims.Default)
```

Configures the maximum number of virtual subscribers allowed to participate in a conference call.

param max_participant The value 0 means that there is no limit. Range: 0 to 10

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.136 TcpAlive

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:TCPalive
```

class TcpAlive

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

get(*ims=<Ims.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:TCPalive
value: bool = driver.configure.data.control.ims.tcpAlive.get(ims = repcap.Ims.
↪Default)
```

Selects whether the IMS server sends TCP keep-alive messages or not.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return tcp_keep: OFF | ON

set(*tcp_keep: bool*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:TCPalive
driver.configure.data.control.ims.tcpAlive.set(tcp_keep = False, ims = repcap.
↪Ims.Default)
```

Selects whether the IMS server sends TCP keep-alive messages or not.

param tcp_keep OFF | ON

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.137 Threshold

class Threshold

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

Cloning the Group

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

Subgroups

7.2.1.1.3.138 Value

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:THReshold:VALue
```

class Value

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

get(*ims=<Ims.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:THReshold:VALue
value: int = driver.configure.data.control.ims.threshold.value.get(ims = repcap.
↪Ims.Default)
```

Configures a threshold for the usage of UDP (below threshold) and TCP (above threshold) . The setting is only relevant for method RsCmwDau.Configure.Data.Control.Ims.Transport.Selection.set CUSTom.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return threshold_value: Number of characters per SIP message Range: 1 to 65535

set(*threshold_value: int*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:THReshold:VALue
driver.configure.data.control.ims.threshold.value.set(threshold_value = 1, ims␣
↪= repcap.Ims.Default)
```

Configures a threshold for the usage of UDP (below threshold) and TCP (above threshold) . The setting is only relevant for method RsCmwDau.Configure.Data.Control.Ims.Transport.Selection.set CUSTom.

param threshold_value Number of characters per SIP message Range: 1 to 65535

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.139 Transport

class Transport

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

Cloning the Group

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

Subgroups

7.2.1.1.3.140 Selection

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:TRANsport:SELection
```

class Selection

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.TransportSel

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:TRANsport:SELection
value: enums.TransportSel = driver.configure.data.control.ims.transport.
↪selection.get(ims = repcap.Ims.Default)
```

Configures whether TCP or UDP is used by the internal IMS server.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return transport_selection: DEFault | TCP | UDP | CUSTom DEFault: Fixed threshold as defined in RFC 3261 TCP: Only TCP is used UDP: Only UDP is used CUSTom: UDP for short messages, TCP for long messages, threshold configurable via method RsCmwDau.Configure.Data.Control.Ims.Threshold.Value.set

set(*transport_selection: RsCmwDau.enums.TransportSel*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:TRANsport:SELection
driver.configure.data.control.ims.transport.selection.set(transport_selection =␣
↪enums.TransportSel.CUSTom, ims = repcap.Ims.Default)
```

Configures whether TCP or UDP is used by the internal IMS server.

param transport_selection DEFault | TCP | UDP | CUSTom DEFault: Fixed threshold as defined in RFC 3261 TCP: Only TCP is used UDP: Only UDP is used CUSTom: UDP for short messages, TCP for long messages, threshold configurable via method RsCmwDau.Configure.Data.Control.Ims.Threshold.Value.set

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.141 Mobile<Profile>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.data.control.ims.mobile.repcap_profile_get()
driver.configure.data.control.ims.mobile.repcap_profile_set(repcap.Profile.Nr1)
```

class Mobile

Mobile commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Profile, default value after init: Profile.Nr1

Cloning the Group

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

Subgroups

7.2.1.1.3.142 DeRegister

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:MOBile<Profile>:DERegister
```

class DeRegister

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

set(*ims=<Ims.Default: -1>*, *profile=<Profile.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:MOBile<UE>:DERegister
driver.configure.data.control.ims.mobile.deRegister.set(ims = repcap.Ims.
↪Default, profile = repcap.Profile.Default)
```

Deregisters a registered subscriber from the IMS server.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

set_with_opc(*ims=<Ims.Default: -1>*, *profile=<Profile.Default: -1>*) → None

7.2.1.1.3.143 Susage

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUSage
```

class Susage

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.SourceInt

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUSage
value: enums.SourceInt = driver.configure.data.control.ims.susage.get(ims =␣
↪repcap.Ims.Default)
```

Selects whether the internal IMS server of the DAU or an external IMS network is used for IMS services.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return server_usage: INTernal | EXTernal

set(*server_usage: RsCmwDau.enums.SourceInt*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUSage
driver.configure.data.control.ims.susage.set(server_usage = enums.SourceInt.
↪EXTernal, ims = repcap.Ims.Default)
```

Selects whether the internal IMS server of the DAU or an external IMS network is used for IMS services.

param server_usage INTernal | EXTernal

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.144 Intern

class Intern

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

Cloning the Group

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

Subgroups

7.2.1.1.3.145 Pcscf

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:INTern:PCSCf:ATYPe
```

class Pcscf

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

get_atype() → RsCmwDau.enums.AddressType

```
# SCPI: CONFigure:DATA:CONTrol:IMS:INTern:PCSCf:ATYPe
value: enums.AddressType = driver.configure.data.control.ims.intern.pcscf.get_
↪atype()
```

No command help available

return addr_type: No help available

set_atype(*addr_type: RsCmwDau.enums.AddressType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:INTern:PCSCf:ATYPe
driver.configure.data.control.ims.intern.pcscf.set_atype(addr_type = enums.
↪AddressType.IPVFour)
```

No command help available

param addr_type No help available

7.2.1.1.3.146 Extern

class Extern

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

Cloning the Group

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

Subgroups

7.2.1.1.3.147 Pcscf

class Pcscf

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

Cloning the Group

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

Subgroups

7.2.1.1.3.148 Address

class Address

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.extern.pcscf.address.clone()
```

Subgroups

7.2.1.1.3.149 IpvFour

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:EXTern:PCSCf:ADDRess:IPVFour
```

class IpvFour

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

get(*ims=<Ims.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:EXTern:PCSCf:ADDRess:IPVFour
value: str = driver.configure.data.control.ims.extern.pcscf.address.ipvFour.
↪get(ims = repcap.Ims.Default)
```

Specifies the IPv4 address of an external P-CSCF.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return ip_v_4_address: IPv4 address string

set(*ip_v_4_address: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:EXTern:PCSCf:ADDRess:IPVFour
driver.configure.data.control.ims.extern.pcscf.address.ipvFour.set(ip_v_4_
↪address = '1', ims = repcap.Ims.Default)
```

Specifies the IPv4 address of an external P-CSCF.

param ip_v_4_address IPv4 address string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.150 IpvSix

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:EXTern:PCSCf:ADDRess:IPVSix
```

class IpvSix

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

get(*ims=<Ims.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:EXTern:PCSCf:ADDRess:IPVSix
value: str = driver.configure.data.control.ims.extern.pcscf.address.ipvSix.
↪get(ims = repcap.Ims.Default)
```

Specifies the IPv6 address of an external P-CSCF.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return ip_v_6_address: IPv6 address string

set(*ip_v_6_address: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:EXTern:PCSCf:ADDRess:IPVSix
driver.configure.data.control.ims.extern.pcscf.address.ipvSix.set(ip_v_6_
↪address = '1', ims = repcap.Ims.Default)
```

Specifies the IPv6 address of an external P-CSCF.

param ip_v_6_address IPv6 address string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.151 Uauthentication

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:UAUThentic:PUID
CONFigure:DATA:CONTrol:IMS:UAUThentic:KEY
CONFigure:DATA:CONTrol:IMS:UAUThentic:RAND
CONFigure:DATA:CONTrol:IMS:UAUThentic:ALGorithm
CONFigure:DATA:CONTrol:IMS:UAUThentic:AMF
CONFigure:DATA:CONTrol:IMS:UAUThentic:AKAVersion
CONFigure:DATA:CONTrol:IMS:UAUThentic:KTYPe
CONFigure:DATA:CONTrol:IMS:UAUThentic:AOP
CONFigure:DATA:CONTrol:IMS:UAUThentic:AOPC
```

(continues on next page)

(continued from previous page)

```
CONFigure:DATA:CONTrol:IMS:UAUThentic:RESLength
CONFigure:DATA:CONTrol:IMS:UAUThentic
```

class Uauthentication

Uauthentication commands group definition. 14 total commands, 1 Sub-groups, 11 group commands

get_aka_version() → RsCmwDau.enums.AkaVersion

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AKAVersion
value: enums.AkaVersion = driver.configure.data.control.ims.uauthentication.get_
↪aka_version()
```

No command help available

return aka_version: No help available

get_algorithm() → RsCmwDau.enums.AuthAlgorithm

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:ALGorithm
value: enums.AuthAlgorithm = driver.configure.data.control.ims.uauthentication.
↪get_algorithm()
```

No command help available

return algorithm: No help available

get_amf() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AMF
value: str = driver.configure.data.control.ims.uauthentication.get_amf()
```

No command help available

return amf: No help available

get_aop() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AOP
value: str = driver.configure.data.control.ims.uauthentication.get_aop()
```

No command help available

return aop: No help available

get_aopc() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AOPC
value: str = driver.configure.data.control.ims.uauthentication.get_aopc()
```

No command help available

return aopc: No help available

get_key() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:KEY
value: str = driver.configure.data.control.ims.uauthentication.get_key()
```

No command help available

return key: No help available

get_ktype() → RsCmwDau.enums.KeyType

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:KTYPe
value: enums.KeyType = driver.configure.data.control.ims.uauthentication.get_
↪ktype()
```

No command help available

return key_type: No help available

get_puid() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:PUID
value: str = driver.configure.data.control.ims.uauthentication.get_puid()
```

No command help available

return private_user_id: No help available

get_rand() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:RAND
value: str = driver.configure.data.control.ims.uauthentication.get_rand()
```

No command help available

return rand: No help available

get_res_length() → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:RESLength
value: int = driver.configure.data.control.ims.uauthentication.get_res_length()
```

No command help available

return res_length: No help available

get_value() → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic
value: bool = driver.configure.data.control.ims.uauthentication.get_value()
```

No command help available

return uauthentic: No help available

set_aka_version(*aka_version: RsCmwDau.enums.AkaVersion*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AKAVersion
driver.configure.data.control.ims.uauthentication.set_aka_version(aka_version =␣
↪enums.AkaVersion.AKA1)
```

No command help available

param aka_version No help available

set_algorithm(*algorithm: RsCmwDau.enums.AuthAlgorithm*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:ALGorithm
driver.configure.data.control.ims.uauthentication.set_algorithm(algorithm =␣
↪enums.AuthAlgorithm.MILenage)
```

No command help available

param algorithm No help available

set_amf(*amf: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AMF
driver.configure.data.control.ims.uauthentication.set_amf(amf = r1)
```

No command help available

param amf No help available

set_aop(*aop: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AOP
driver.configure.data.control.ims.uauthentication.set_aop(aop = r1)
```

No command help available

param aop No help available

set_aopc(*aopc: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:AOPC
driver.configure.data.control.ims.uauthentication.set_aopc(aopc = r1)
```

No command help available

param aopc No help available

set_key(*key: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:KEY
driver.configure.data.control.ims.uauthentication.set_key(key = r1)
```

No command help available

param key No help available

set_ktype(*key_type: RsCmwDau.enums.KeyType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:KTYPe
driver.configure.data.control.ims.uauthentication.set_ktype(key_type = enums.
↪KeyType.OP)
```

No command help available

param key_type No help available

set_puid(*private_user_id: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:PUID
driver.configure.data.control.ims.uauthentication.set_puid(private_user_id = '1
↪')
```

No command help available

param private_user_id No help available

set_rand(*rand: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:RAND
driver.configure.data.control.ims.uauthentication.set_rand(rand = r1)
```

No command help available

param rand No help available

set_res_length(*res_length: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:RESLength
driver.configure.data.control.ims.uauthentication.set_res_length(res_length = 1)
```

No command help available

param res_length No help available

set_value(*uauthentic: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic
driver.configure.data.control.ims.uauthentication.set_value(uauthentic = False)
```

No command help available

param uauthentic No help available

Cloning the Group

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

Subgroups

7.2.1.1.3.152 IpSec

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec:IALGorithm
CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec:EALGorithm
CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec
```

class IpSec

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

get_ealgorithm() → RsCmwDau.enums.IpSecEalgorithm

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec:EALGorithm
value: enums.IpSecEalgorithm = driver.configure.data.control.ims.
↪uauthentication.ipSec.get_ealgorithm()
```

No command help available

return ip_sece_alg: No help available

get_ialgorithm() → RsCmwDau.enums.IpSecIalgorithm

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec:IALGorithm
value: enums.IpSecIalgorithm = driver.configure.data.control.ims.
↪uauthentication.ipSec.get_ialgorithm()
```

No command help available

return ip_sec_ialgorithm: No help available

get_value() → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec
value: bool = driver.configure.data.control.ims.uauthentication.ipSec.get_
↪value()
```

No command help available

return ip_sec: No help available

set_ealgorithm(*ip_sece_alg: RsCmwDau.enums.IpSecEalgorithm*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec:EALGorithm
driver.configure.data.control.ims.uauthentication.ipSec.set_ealgorithm(ip_sece_
↪alg = enums.IpSecEalgorithm.AES)
```

No command help available

param ip_sece_alg No help available

set_ialgorithm(*ip_sec_ialgorithm: RsCmwDau.enums.IpSecIalgorithm*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec:IALGorithm
driver.configure.data.control.ims.uauthentication.ipSec.set_ialgorithm(ip_sec_
↪ialgorithm = enums.IpSecIalgorithm.AUTO)
```

No command help available

param ip_sec_ialgorithm No help available

set_value(*ip_sec: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:UAUThentic:IPSec
driver.configure.data.control.ims.uauthentication.ipSec.set_value(ip_sec =␣
↪False)
```

No command help available

param ip_sec No help available

7.2.1.1.3.153 Clean

class Clean

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

Cloning the Group

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

Subgroups

7.2.1.1.3.154 General

class General

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

Cloning the Group

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

Subgroups

7.2.1.1.3.155 Info

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:CLEan:GENeral:INFO
```

class Info

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

set(*ims*=<*Ims.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:CLEan:GENeral:INFO
driver.configure.data.control.ims.clean.general.info.set(ims = repcap.Ims.
↪Default)
```

Clears the 'General IMS Info' area.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims*=<*Ims.Default: -1*>) → None

7.2.1.1.3.156 Subscriber<Subscriber>

RepCap Settings

```
# Range: Nr1 .. Nr5
rc = driver.configure.data.control.ims.subscriber.repcap_subscriber_get()
driver.configure.data.control.ims.subscriber.repcap_subscriber_set(repcap.Subscriber.Nr1)
```

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:DELete
```

class Subscriber

Subscriber commands group definition. 16 total commands, 9 Sub-groups, 1 group commands Repeated Capability: Subscriber, default value after init: Subscriber.Nr1

delete(*ims*=<*Ims.Default: -1*>, *subscriber*=<*Subscriber.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:DELete
driver.configure.data.control.ims.subscriber.delete(ims = repcap.Ims.Default,␣
↪subscriber = repcap.Subscriber.Default)
```

(continues on next page)

(continued from previous page)

Deletes the subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

delete_with_opc(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

Cloning the Group

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

Subgroups

7.2.1.1.3.157 Impu

class Impu

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

Cloning the Group

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

Subgroups

7.2.1.1.3.158 Header

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:IMPU:HEADer
```

class Header

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → RsCmwDau.enums.PauHeader

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:IMPU:HEADer
value: enums.PauHeader = driver.configure.data.control.ims.subscriber.impu.
↪header.get(ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)


    INTRO_CMD_HELP: Selects which public IDs are sent to the DUT in the PAU␣
↪header of the 200/OK response to REGISTER messages:
```

(continues on next page)

(continued from previous page)

```
    - Configured: public user IDs configured in the subscriber settings
    - Registered: ID sent by the DUT in the 'from' header of the REGISTER␣
↪message
    - Generated: automatically generated ID starting with 'tel:555'

    :param ims: optional repeated capability selector. Default value: Ix1␣
↪(settable in the interface 'Ims')
    :param subscriber: optional repeated capability selector. Default value:␣
↪Nr1 (settable in the interface 'Subscriber')
    :return: pau_header: CONRege | RECoge | CORE | RECN | COGE | REGE | CONF |␣
↪REGD CONRege: configured, registered, generated RECoge: registered,␣
↪configured, generated CORE: configured, registered RECN: registered,␣
↪configured COGE: configured, generated REGE: registered, generated CONF:␣
↪configured REGD: registered
```

set(*pau_header: RsCmwDau.enums.PauHeader*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:IMPU:HEADer
driver.configure.data.control.ims.subscriber.impu.header.set(pau_header = enums.
↪PauHeader.COGE, ims = repcap.Ims.Default, subscriber = repcap.Subscriber.
↪Default)


    INTRO_CMD_HELP: Selects which public IDs are sent to the DUT in the PAU␣
↪header of the 200/OK response to REGISTER messages:

    - Configured: public user IDs configured in the subscriber settings
    - Registered: ID sent by the DUT in the 'from' header of the REGISTER␣
↪message
    - Generated: automatically generated ID starting with 'tel:555'

    :param pau_header: CONRege | RECoge | CORE | RECN | COGE | REGE | CONF |␣
↪REGD CONRege: configured, registered, generated RECoge: registered,␣
↪configured, generated CORE: configured, registered RECN: registered,␣
↪configured COGE: configured, generated REGE: registered, generated CONF:␣
↪configured REGD: registered
    :param ims: optional repeated capability selector. Default value: Ix1␣
↪(settable in the interface 'Ims')
    :param subscriber: optional repeated capability selector. Default value:␣
↪Nr1 (settable in the interface 'Subscriber')
```

7.2.1.1.3.159 ChatQci

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:CHATqci
```

class ChatQci

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:CHATqci
value: int = driver.configure.data.control.ims.subscriber.chatQci.get(ims =␣
↪repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the QCI used by the DUT for RCS message transfer.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return chat_qci: Range: 0 to 255

set(*chat_qci: int*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:CHATqci
driver.configure.data.control.ims.subscriber.chatQci.set(chat_qci = 1, ims =␣
↪repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the QCI used by the DUT for RCS message transfer.

param chat_qci Range: 0 to 255

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.160 PrivateId

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:PRIVateid
```

class PrivateId

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:PRIVateid
value: str = driver.configure.data.control.ims.subscriber.privateId.get(ims =␣
↪repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the private user ID of the subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return private_id: Private user ID as string

set(*private_id: str*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:PRIVateid
driver.configure.data.control.ims.subscriber.privateId.set(private_id = '1',␣
↪ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the private user ID of the subscriber profile number <s>.

param private_id Private user ID as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.161 Authentication

class Authentication

Authentication commands group definition. 5 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.162 Scheme

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:AUTHenticati:SCHeme
```

class Scheme

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → RsCmwDau.enums.AuthScheme

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:SCHeme
value: enums.AuthScheme = driver.configure.data.control.ims.subscriber.
↪authentication.scheme.get(ims = repcap.Ims.Default, subscriber = repcap.
↪Subscriber.Default)
```

Specifies whether authentication is performed for the subscriber profile number <s> and selects the authentication and key agreement version to be used.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return auth_scheme: AKA1 | AKA2 | NOAuthentic AKA1: authentication with AKA version 1 AKA2: authentication with AKA version 2 NOAuthentic: no authentication

set(*auth_scheme: RsCmwDau.enums.AuthScheme*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:SCHeme
driver.configure.data.control.ims.subscriber.authentication.scheme.set(auth_
↪scheme = enums.AuthScheme.AKA1, ims = repcap.Ims.Default, subscriber = repcap.
↪Subscriber.Default)
```

Specifies whether authentication is performed for the subscriber profile number <s> and selects the authentication and key agreement version to be used.

param auth_scheme AKA1 | AKA2 | NOAuthentic AKA1: authentication with AKA version 1 AKA2: authentication with AKA version 2 NOAuthentic: no authentication

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.163 Algorithm

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:AUTHenticati:ALGorithm
```

class Algorithm

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → RsCmwDau.enums.AuthAlgorithm

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:ALGorithm
value: enums.AuthAlgorithm = driver.configure.data.control.ims.subscriber.
↪authentication.algorithm.get(ims = repcap.Ims.Default, subscriber = repcap.
↪Subscriber.Default)
```

(continues on next page)

(continued from previous page)

Specifies which algorithm set is used for the subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return auth_key_gen_alg: XOR | MILenage

set(*auth_key_gen_alg: RsCmwDau.enums.AuthAlgorithm*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:ALGorithm
driver.configure.data.control.ims.subscriber.authentication.algorithm.set(auth_
↪key_gen_alg = enums.AuthAlgorithm.MILenage, ims = repcap.Ims.Default,␣
↪subscriber = repcap.Subscriber.Default)
```

Specifies which algorithm set is used for the subscriber profile number <s>.

param auth_key_gen_alg XOR | MILenage

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.164 Key

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:AUTHenticati:KEY
```

class Key

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:KEY
value: str = driver.configure.data.control.ims.subscriber.authentication.key.
↪get(ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Defines the authentication key K for the subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return auth_key: Key as 32-digit hexadecimal number A query returns a string. A setting supports the string format and the hexadecimal format (#H…) .

set(*auth_key: str*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:KEY
driver.configure.data.control.ims.subscriber.authentication.key.set(auth_key =␣
↪r1, ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Defines the authentication key K for the subscriber profile number <s>.

param auth_key Key as 32-digit hexadecimal number A query returns a string. A setting supports the string format and the hexadecimal format (#H…) .

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.165 Amf

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:AUTHenticati:AMF
```

class Amf

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:AMF
value: str = driver.configure.data.control.ims.subscriber.authentication.amf.
↪get(ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the authentication management field (AMF) for the subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return auth_amf: AMF as four-digit hexadecimal number A query returns a string. A setting supports the string format and the hexadecimal format (#H…) .

set(*auth_amf: str*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:AMF
driver.configure.data.control.ims.subscriber.authentication.amf.set(auth_amf =␣
↪r1, ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the authentication management field (AMF) for the subscriber profile number <s>.

param auth_amf AMF as four-digit hexadecimal number A query returns a string. A setting supports the string format and the hexadecimal format (#H...) .

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.166 Opc

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:AUTHenticati:OPC
```

class Opc

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:OPC
value: str = driver.configure.data.control.ims.subscriber.authentication.opc.
↪get(ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the key OPc for the subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return auth_opc: Key as 32-digit hexadecimal number A query returns a string. A setting supports the string format and the hexadecimal format (#H...) .

set(*auth_opc: str*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:AUTHenticati:OPC
driver.configure.data.control.ims.subscriber.authentication.opc.set(auth_opc =␣
↪r1, ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Specifies the key OPc for the subscriber profile number <s>.

param auth_opc Key as 32-digit hexadecimal number A query returns a string. A setting supports the string format and the hexadecimal format (#H...) .

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.167 ResLength

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:RESLength
```

class ResLength

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:RESLength
value: int = driver.configure.data.control.ims.subscriber.resLength.get(ims =␣
↪repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return res_length: No help available

set(*res_length: int*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:RESLength
driver.configure.data.control.ims.subscriber.resLength.set(res_length = 1, ims␣
↪= repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

No command help available

param res_length No help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.168 IpSec

class IpSec

IpSec commands group definition. 3 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.169 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:IPSec:ENABle
```

class Enable

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:IPSec:ENABle
value: bool = driver.configure.data.control.ims.subscriber.ipSec.enable.get(ims␣
↪= repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Enables or disables support of the IP security mechanisms by the IMS server for subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return ip_sec: OFF | ON

set(*ip_sec: bool*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:IPSec:ENABle
driver.configure.data.control.ims.subscriber.ipSec.enable.set(ip_sec = False,␣
↪ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default)
```

Enables or disables support of the IP security mechanisms by the IMS server for subscriber profile number <s>.

param ip_sec OFF | ON

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.170 Algorithm

class Algorithm

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.subscriber.ipSec.algorithm.clone()
```

Subgroups

7.2.1.1.3.171 Integrity

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:IPSec:ALGorithm:INTegrity
```

class Integrity

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → RsCmwDau.enums.IpSecIalgorithm

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:IPSec:ALGorithm:INTegrity
value: enums.IpSecIalgorithm = driver.configure.data.control.ims.subscriber.
↪ipSec.algorithm.integrity.get(ims = repcap.Ims.Default, subscriber = repcap.
↪Subscriber.Default)
```

Selects an integrity protection algorithm for subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return integrity_alg: HMMD | HMSH | AUTO HMMD: HMAC-MD5-96 HMSH: HMAC-SHA-1-96 AUTO: as indicated in REGISTER message

set(*integrity_alg: RsCmwDau.enums.IpSecIalgorithm*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:IPSec:ALGorithm:INTegrity
driver.configure.data.control.ims.subscriber.ipSec.algorithm.integrity.
↪set(integrity_alg = enums.IpSecIalgorithm.AUTO, ims = repcap.Ims.Default,␣
↪subscriber = repcap.Subscriber.Default)
```

Selects an integrity protection algorithm for subscriber profile number <s>.

param integrity_alg HMMD | HMSH | AUTO HMMD: HMAC-MD5-96 HMSH: HMAC-SHA-1-96 AUTO: as indicated in REGISTER message

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.172 Encryption

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:IPSec:ALGorithm:ENCRyption
```

class Encryption

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

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → RsCmwDau.enums.IpSecEalgorithm

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:IPSec:ALGorithm:ENCRyption
value: enums.IpSecEalgorithm = driver.configure.data.control.ims.subscriber.
↪ipSec.algorithm.encryption.get(ims = repcap.Ims.Default, subscriber = repcap.
↪Subscriber.Default)
```

Selects an encryption algorithm for subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return encryption_alg: DES | AES | NOC | AUTO DES: DES-EDE3-CBC AES: AES-CBC NOC: NULL, no encryption AUTO: as indicated in REGISTER message

set(*encryption_alg: RsCmwDau.enums.IpSecEalgorithm*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>
↪:IPSec:ALGorithm:ENCRyption
driver.configure.data.control.ims.subscriber.ipSec.algorithm.encryption.
↪set(encryption_alg = enums.IpSecEalgorithm.AES, ims = repcap.Ims.Default,␣
↪subscriber = repcap.Subscriber.Default)
```

Selects an encryption algorithm for subscriber profile number <s>.

param encryption_alg DES | AES | NOC | AUTO DES: DES-EDE3-CBC AES: AES-CBC NOC: NULL, no encryption AUTO: as indicated in REGISTER message

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.173 PublicUserId<UserId>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.subscriber.publicUserId.repcap_userId_get()
driver.configure.data.control.ims.subscriber.publicUserId.repcap_userId_set(repcap.
↪UserId.Ix1)
```

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber<Subscriber>:PUBLicuserid<UserId>
```

class PublicUserId

PublicUserId commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: UserId, default value after init: UserId.Ix1

get(*ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*, *userId=<UserId.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:PUBLicuserid
↪<Index>
value: str = driver.configure.data.control.ims.subscriber.publicUserId.get(ims␣
↪= repcap.Ims.Default, subscriber = repcap.Subscriber.Default, userId = repcap.
↪UserId.Default)
```

Defines public user ID number <Index> for the subscriber profile number <s>.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param userId optional repeated capability selector. Default value: Ix1 (settable in the interface 'PublicUserId')

return public_user_ids: Public user ID as string

set(*public_user_ids: str*, *ims=<Ims.Default: -1>*, *subscriber=<Subscriber.Default: -1>*, *userId=<UserId.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber<Subscriber>:PUBLicuserid
↪<Index>
driver.configure.data.control.ims.subscriber.publicUserId.set(public_user_ids =
↪'1', ims = repcap.Ims.Default, subscriber = repcap.Subscriber.Default, userId␣
↪= repcap.UserId.Default)
```

Defines public user ID number <Index> for the subscriber profile number <s>.

param public_user_ids Public user ID as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

param userId optional repeated capability selector. Default value: Ix1 (settable in the interface 'PublicUserId')

Cloning the Group

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

7.2.1.1.3.174 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber:ADD
```

class Add

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

set(*ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber:ADD
driver.configure.data.control.ims.subscriber.add.set(ims = repcap.Ims.Default)
```

Creates a subscriber profile. See also method RsCmwDau.Configure.Data.Control.Ims.Subscriber.Create.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims=<Ims.Default: -1>*) → None

7.2.1.1.3.175 Create

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:SUBScriber:CREate
```

class Create

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

set(*ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:SUBScriber:CREate
driver.configure.data.control.ims.subscriber.create.set(ims = repcap.Ims.
↪Default)
```

Updates the internal list of subscriber profiles. If your command script adds subscriber profiles, you must insert this command before you can use the new profiles. It is sufficient to insert the command once, after adding the last subscriber profile / before using the profiles.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims=<Ims.Default: -1>*) → None

7.2.1.1.3.176 Pcscf<PcscFnc>

RepCap Settings

```
# Range: Nr1 .. Nr10
rc = driver.configure.data.control.ims.pcscf.repcap_pcscFnc_get()
driver.configure.data.control.ims.pcscf.repcap_pcscFnc_set(repcap.PcscFnc.Nr1)
```

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf<PcscFnc>:DELete
```

class Pcscf

Pcscf commands group definition. 9 total commands, 8 Sub-groups, 1 group commands Repeated Capability: PcscFnc, default value after init: PcscFnc.Nr1

delete(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:DELete
driver.configure.data.control.ims.pcscf.delete(ims = repcap.Ims.Default,␣
↪pcscFnc = repcap.PcscFnc.Default)
```

Deletes the P-CSCF profile number {p}.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

delete_with_opc(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

Cloning the Group

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

Subgroups

7.2.1.1.3.177 IpAddress

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf<PcscFnc>:IPADdress
```

class IpAddress

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

get(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:IPADdress
value: str = driver.configure.data.control.ims.pcscf.ipAddress.get(ims = repcap.
↪Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines the IP address of the P-CSCF number {p}.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return ip_address: IPv4 or IPv6 address string Example: '172.22.1.201' or 'fd01:cafe::1/64'

set(*ip_address: str*, *ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:IPADdress
driver.configure.data.control.ims.pcscf.ipAddress.set(ip_address = '1', ims =␣
↪repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines the IP address of the P-CSCF number {p}.

param ip_address IPv4 or IPv6 address string Example: '172.22.1.201' or 'fd01:cafe::1/64'

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.178 Behaviour

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf<PcscFnc>:BEHaviour
```

class Behaviour

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

get(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → RsCmwDau.enums.BehaviourB

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:BEHaviour
value: enums.BehaviourB = driver.configure.data.control.ims.pcscf.behaviour.
↪get(ims = repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines the behavior of the P-CSCF number {p} when it receives a SIP message from the DUT.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return behaviour: NORMal | FAILure NORMal: normal behavior FAILure: return failure code

set(*behaviour: RsCmwDau.enums.BehaviourB*, *ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:BEHaviour
driver.configure.data.control.ims.pcscf.behaviour.set(behaviour = enums.
↪BehaviourB.FAILure, ims = repcap.Ims.Default, pcscFnc = repcap.PcscFnc.
↪Default)
```

Defines the behavior of the P-CSCF number {p} when it receives a SIP message from the DUT.

param behaviour NORMal | FAILure NORMal: normal behavior FAILure: return failure code

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.179 FailureCode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf<PcscFnc>:FAILurecode
```

class FailureCode

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

get(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:FAILurecode
value: int = driver.configure.data.control.ims.pcscf.failureCode.get(ims =␣
↪repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines a failure code for the P-CSCF number {p}, behavior = FAIL.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return failure_code: BADRequest | FORBidden | NOTFound | INTerror | UNAVailable | BUSYeveryw BADRequest: '400 Bad Request' FORBidden: '403 Forbidden' NOTFound: '404 Not Found' INTerror: '500 Server Internal Error' UNAVailable: '503 Service Unavailable' BUSYeveryw: '600 Busy Everywhere'

set(*failure_code: int*, *ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:FAILurecode
driver.configure.data.control.ims.pcscf.failureCode.set(failure_code = 1, ims =␣
↪repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines a failure code for the P-CSCF number {p}, behavior = FAIL.

param failure_code BADRequest | FORBidden | NOTFound | INTerror | UNAVailable | BUSYeveryw BADRequest: '400 Bad Request' FORBidden: '403 Forbidden' NOTFound: '404 Not Found' INTerror: '500 Server Internal Error' UNAVailable: '503 Service Unavailable' BUSYeveryw: '600 Busy Everywhere'

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.180 RetryAfter

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf<PcscFnc>:RETRyafter
```

class RetryAfter

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

get(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:RETRyafter
value: int = driver.configure.data.control.ims.pcscf.retryAfter.get(ims =␣
↪repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines the contents of the 'Retry After' header field for the P-CSCF number {p}, behavior = FAIL.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

return retry_after: Unit: s

set(*retry_after: int*, *ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:RETRyafter
driver.configure.data.control.ims.pcscf.retryAfter.set(retry_after = 1, ims =␣
↪repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines the contents of the 'Retry After' header field for the P-CSCF number {p}, behavior = FAIL.

param retry_after Unit: s

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.181 RegExp

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf<PcscFnc>:REGexp
```

class RegExp

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

class RegExpStruct

Structure for setting input parameters. Fields:

- Reg_Exp_Min: int: Minimum acceptable expiration time Unit: s
- Reg_Exp_Default: int: Default value, used if the DUT does not suggest an expiration time Unit: s
- Reg_Exp_Max: int: Maximum acceptable expiration time Unit: s

get(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → RegExpStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:REGexp
value: RegExpStruct = driver.configure.data.control.ims.pcscf.regExp.get(ims =␣
↪repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines registration expiration times for the P-CSCF number {p}.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.Pcscf_.RegExp.RegExp.RegExpStruct*, *ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:REGexp
driver.configure.data.control.ims.pcscf.regExp.set(value = [PROPERTY_STRUCT_
↪NAME](), ims = repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines registration expiration times for the P-CSCF number {p}.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.182 SubExp

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf<PcscFnc>:SUBexp
```

class SubExp

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

class SubExpStruct

Structure for setting input parameters. Fields:

- Subs_Exp_Min: int: Minimum acceptable expiration time Unit: s
- Subs_Exp_Default: int: Default value, used if the DUT does not suggest an expiration time Unit: s
- Subs_Exp_Max: int: Maximum acceptable expiration time Unit: s

get(*ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → SubExpStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:SUBexp
value: SubExpStruct = driver.configure.data.control.ims.pcscf.subExp.get(ims =␣
↪repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines subscription expiration times for the P-CSCF number {p}.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.Pcscf_.SubExp.SubExp.SubExpStruct*, *ims=<Ims.Default: -1>*, *pcscFnc=<PcscFnc.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf<Pcscf>:SUBexp
driver.configure.data.control.ims.pcscf.subExp.set(value = [PROPERTY_STRUCT_
↪NAME](), ims = repcap.Ims.Default, pcscFnc = repcap.PcscFnc.Default)
```

Defines subscription expiration times for the P-CSCF number {p}.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

7.2.1.1.3.183 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf:ADD
```

class Add

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

set(*ims*=<*Ims.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf:ADD
driver.configure.data.control.ims.pcscf.add.set(ims = repcap.Ims.Default)
```

Creates a P-CSCF profile. See also method RsCmwDau.Configure.Data.Control.Ims.Pcscf.Create.set

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims*=<*Ims.Default: -1*>) → None

7.2.1.1.3.184 Create

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:PCSCf:CREate
```

class Create

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

set(*ims*=<*Ims.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:PCSCf:CREate
driver.configure.data.control.ims.pcscf.create.set(ims = repcap.Ims.Default)
```

Updates the internal list of P-CSCF profiles. If your command script adds P-CSCF profiles, you must insert this command before you can use the new profiles. It is sufficient to insert the command once, after adding the last P-CSCF profile / before using the profiles.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims*=<*Ims.Default: -1*>) → None

7.2.1.1.3.185 Update

class Update
Update commands group definition. 33 total commands, 8 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.186 Rcs

class Rcs
Rcs commands group definition. 4 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.187 Chat

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.rcs.chat.clone()
```

Subgroups

7.2.1.1.3.188 Perform

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:RCS:CHAT:PERForm
```

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

set(*ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:RCS:CHAT:PERForm
driver.configure.data.control.ims.update.rcs.chat.perform.set(ims = repcap.Ims.
↪Default)
```

Initiates a chat message transfer to the DUT.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims=<Ims.Default: -1>*) → None

7.2.1.1.3.189 Text

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:RCS:CHAT:TEXT
```

class Text

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

get(*ims=<Ims.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:RCS:CHAT:TEXT
value: str = driver.configure.data.control.ims.update.rcs.chat.text.get(ims =␣
↪repcap.Ims.Default)
```

Defines a message text to be sent to the DUT via an established chat session. Initiate the message transfer via method RsCmwDau.Configure.Data.Control.Ims.Update.Rcs.Chat.Perform.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return text: Message as string

set(*text: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:RCS:CHAT:TEXT
driver.configure.data.control.ims.update.rcs.chat.text.set(text = '1', ims =␣
↪repcap.Ims.Default)
```

Defines a message text to be sent to the DUT via an established chat session. Initiate the message transfer via method RsCmwDau.Configure.Data.Control.Ims.Update.Rcs.Chat.Perform.set.

param text Message as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.190 Idle

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.rcs.idle.clone()
```

Subgroups

7.2.1.1.3.191 Ntfcn

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:RCS:IDLE:NTFCn
```

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

set(*ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:RCS:IDLE:NTFCn
driver.configure.data.control.ims.update.rcs.idle.ntfcn.set(ims = repcap.Ims.
↪Default)
```

Send an 'idle' notification to the DUT as 'isComposing' status message.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims=<Ims.Default: -1>*) → None

7.2.1.1.3.192 CompSng

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.rcs.compSng.clone()
```

Subgroups

7.2.1.1.3.193 Ntfcn

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:RCS:COMPsng:NTFCn
```

class Ntfcn

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

set(*ims*=<*Ims.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:RCS:COMPsng:NTFCn
driver.configure.data.control.ims.update.rcs.compSng.ntfcn.set(ims = repcap.Ims.
↪Default)
```

Send an 'active' notification to the DUT as 'isComposing' status message.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims*=<*Ims.Default: -1*>) → None

7.2.1.1.3.194 Inband

class Inband

Inband commands group definition. 6 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.195 Perform

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:INBand:PERForm
```

class Perform

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

set(*ims*=<*Ims.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:PERForm
driver.configure.data.control.ims.update.inband.perform.set(ims = repcap.Ims.
↪Default)
```

Initiates an 'inband' call update. The settings configured via the UPDate commands are applied to the call selected via method RsCmwDau.Configure.Data.Control.Ims.Update.Call.Id.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims*=<*Ims.Default: -1*>) → None

7.2.1.1.3.196 Evs

class Evs

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.inband.evs.clone()
```

Subgroups

7.2.1.1.3.197 Bw

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:INBand:EVS:BW
```

class Bw

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

get(*ims*=<*Ims.Default: -1*>) → RsCmwDau.enums.EvsBw

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:EVS:BW
value: enums.EvsBw = driver.configure.data.control.ims.update.inband.evs.bw.
↪get(ims = repcap.Ims.Default)
```

Configures the EVS codec bandwidth to be requested via CMR.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return bw: NB | IO | WB | SWB | FB | WBCA | SWBCa | NOReq | DEAC IO: AMR-WB IO mode NB: primary, narrowband WB: primary, wideband SWB: primary, super wideband FB: primary, fullband WBCA: primary, WB, channel-aware mode SWBCa: primary, SWB, channel-aware mode NOReq: NO_REQ, no codec rate requirement DEAC: CMR byte removed from header

set(*bw: RsCmwDau.enums.EvsBw*, *ims*=<*Ims.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:EVS:BW
driver.configure.data.control.ims.update.inband.evs.bw.set(bw = enums.EvsBw.
↪DEAC, ims = repcap.Ims.Default)
```

Configures the EVS codec bandwidth to be requested via CMR.

param bw NB | IO | WB | SWB | FB | WBCA | SWBCa | NOReq | DEAC IO: AMR-WB IO mode NB: primary, narrowband WB: primary, wideband SWB: primary, super wideband FB: primary, fullband WBCA: primary, WB, channel-aware mode SWBCa: primary, SWB, channel-aware mode NOReq: NO_REQ, no codec rate requirement DEAC: CMR byte removed from header

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.198 Codec

class Codec

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.inband.evs.codec.clone()
```

Subgroups

7.2.1.1.3.199 Rates

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:INBand:EVS:CODec:RATes
```

class Rates

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

get(*ims*=<*Ims.Default: -1*>) → RsCmwDau.enums.EvsBitrate

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:EVS:CODec:RATes
value: enums.EvsBitrate = driver.configure.data.control.ims.update.inband.evs.
↪codec.rates.get(ims = repcap.Ims.Default)
```

Configures an EVS codec rate or bit rate to be requested via CMR. For BW=DEAC, you cannot set a rate.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return evs_bitrate: NOReq | AW66 | AW885 | AW1265 | AW1425 | AW1585 | AW1825 | AW1985 | AW2305 | AWB2385 | PR59 | PR72 | PR80 | PR96 | P132 | P164 | P244 | P320 | P480 | P640 | P960 | P1280 | SLO2 | SLO3 | SLO5 | SLO7 | SHO2 | SHO3 | SHO5 | SHO7 | WLO2 | WLO3 | WLO5 | WLO7 | WHO2 | WHO3 | WHO5 | WHO7 NOReq No codec rate requirement (NO_REQ) Only for BW=NOReq AW66 to AWB2385 AMR-WB IO mode, 6.6 kbit/s to 23.85 kbit/s Only for BW=IO PR59 to P1280 Primary mode, 5.9 kbit/s to 128.0 kbit/s For BW=NB: PR59 to P244 For BW=WB: PR59 to P1280 For BW=SWB: PR96 to P1280 For BW=FB: P164 to P1280 SLO2 to SLO7 SWB with channel-aware mode, CA-L-O2 to CA-L-O7 Only for BW=SWBCa SHO2 to SHO7

SWB with channel-aware mode, CA-H-O2 to CA-H-O7 Only for BW=SWBCa WLO2 to WLO7 WB with channel-aware mode, CA-L-O2 to CA-L-O7 Only for BW=WBCA WHO2 to WHO7 WB with channel-aware mode, CA-H-O2 to CA-H-O7 Only for BW=WBCA

set(*evs_bitrate: RsCmwDau.enums.EvsBitrate*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:EVS:CODec:RATes
driver.configure.data.control.ims.update.inband.evs.codec.rates.set(evs_bitrate␣
↪= enums.EvsBitrate.AW1265, ims = repcap.Ims.Default)
```

Configures an EVS codec rate or bit rate to be requested via CMR. For BW=DEAC, you cannot set a rate.

param evs_bitrate NOReq | AW66 | AW885 | AW1265 | AW1425 | AW1585 | AW1825 | AW1985 | AW2305 | AWB2385 | PR59 | PR72 | PR80 | PR96 | P132 | P164 | P244 | P320 | P480 | P640 | P960 | P1280 | SLO2 | SLO3 | SLO5 | SLO7 | SHO2 | SHO3 | SHO5 | SHO7 | WLO2 | WLO3 | WLO5 | WLO7 | WHO2 | WHO3 | WHO5 | WHO7 NOReq No codec rate requirement (NO_REQ) Only for BW=NOReq AW66 to AWB2385 AMR-WB IO mode, 6.6 kbit/s to 23.85 kbit/s Only for BW=IO PR59 to P1280 Primary mode, 5.9 kbit/s to 128.0 kbit/s For BW=NB: PR59 to P244 For BW=WB: PR59 to P1280 For BW=SWB: PR96 to P1280 For BW=FB: P164 to P1280 SLO2 to SLO7 SWB with channel-aware mode, CA-L-O2 to CA-L-O7 Only for BW=SWBCa SHO2 to SHO7 SWB with channel-aware mode, CA-H-O2 to CA-H-O7 Only for BW=SWBCa WLO2 to WLO7 WB with channel-aware mode, CA-L-O2 to CA-L-O7 Only for BW=WBCA WHO2 to WHO7 WB with channel-aware mode, CA-H-O2 to CA-H-O7 Only for BW=WBCA

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.200 Repetition

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:INBand:REPetition
```

class Repetition

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.Repetition

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:REPetition
value: enums.Repetition = driver.configure.data.control.ims.update.inband.
↪repetition.get(ims = repcap.Ims.Default)
```

Selects whether a CMR is sent only once or continuously.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return repetition: ENDLess | ONCE

set(*repetition: RsCmwDau.enums.Repetition*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:REPetition
driver.configure.data.control.ims.update.inband.repetition.set(repetition =␣
↪enums.Repetition.ENDLess, ims = repcap.Ims.Default)
```

Selects whether a CMR is sent only once or continuously.

param repetition ENDLess | ONCE

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.201 AmRnb

class AmRnb

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.inband.amRnb.clone()
```

Subgroups

7.2.1.1.3.202 Codec

class Codec

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.inband.amRnb.codec.clone()
```

Subgroups

7.2.1.1.3.203 Rates

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:INBand:AMRNb:CODec:RATes
```

class Rates

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.AmRnbBitrate

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:AMRNb:CODec:RATes
value: enums.AmRnbBitrate = driver.configure.data.control.ims.update.inband.
↪amRnb.codec.rates.get(ims = repcap.Ims.Default)
```

Configures an AMR narrowband codec rate to be requested via CMR.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return am_rnb_bitrate: R475 | R515 | R590 | R670 | R740 | R795 | R1020 | R1220 | NOReq R475 to R1220: 4.75 kbit/s to 12.20 kbit/s NOReq: no codec rate requirement (NO_REQ)

set(*am_rnb_bitrate: RsCmwDau.enums.AmRnbBitrate*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:AMRNb:CODec:RATes
driver.configure.data.control.ims.update.inband.amRnb.codec.rates.set(am_rnb_
↪bitrate = enums.AmRnbBitrate.NOReq, ims = repcap.Ims.Default)
```

Configures an AMR narrowband codec rate to be requested via CMR.

param am_rnb_bitrate R475 | R515 | R590 | R670 | R740 | R795 | R1020 | R1220 | NOReq R475 to R1220: 4.75 kbit/s to 12.20 kbit/s NOReq: no codec rate requirement (NO_REQ)

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.204 AmRwb

class AmRwb

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.inband.amRwb.clone()
```

Subgroups

7.2.1.1.3.205 Codec

class Codec

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.inband.amRwb.codec.clone()
```

Subgroups

7.2.1.1.3.206 Rates

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:INBand:AMRWb:CODec:RATes
```

class Rates

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.AmRwbBitRate

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:AMRWb:CODec:RATes
value: enums.AmRwbBitRate = driver.configure.data.control.ims.update.inband.
↪amRwb.codec.rates.get(ims = repcap.Ims.Default)
```

Configures an AMR wideband codec rate to be requested via CMR.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return am_rwb_bit_rate: R660 | R885 | R1265 | R1425 | R1585 | R1825 | R1985 | R2305 | RA2385 | NOReq R660 to RA2385: 6.60 kbit/s to 23.85 kbit/s NOReq: no codec rate requirement (NO_REQ)

set(*am_rwb_bit_rate: RsCmwDau.enums.AmRwbBitRate*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:INBand:AMRWb:CODec:RATes
driver.configure.data.control.ims.update.inband.amRwb.codec.rates.set(am_rwb_
↪bit_rate = enums.AmRwbBitRate.NOReq, ims = repcap.Ims.Default)
```

Configures an AMR wideband codec rate to be requested via CMR.

param am_rwb_bit_rate R660 | R885 | R1265 | R1425 | R1585 | R1825 | R1985 | R2305 | RA2385 | NOReq R660 to RA2385: 6.60 kbit/s to 23.85 kbit/s NOReq: no codec rate requirement (NO_REQ)

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.207 Call

class Call

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

Cloning the Group

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

Subgroups

7.2.1.1.3.208 Event

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:CALL:EVENt
```

class Event

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.UpdateCallEvent

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:CALL:EVENt
value: enums.UpdateCallEvent = driver.configure.data.control.ims.update.call.
↪event.get(ims = repcap.Ims.Default)
```

Puts a call on hold or resumes a call that has been put on hold. To select the call, use method RsCmwDau.Configure.Data. Control.Ims.Update.Call.Id.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return update_call_event: HOLD | RESume

set(*update_call_event: RsCmwDau.enums.UpdateCallEvent*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:CALL:EVENt
driver.configure.data.control.ims.update.call.event.set(update_call_event =␣
↪enums.UpdateCallEvent.HOLD, ims = repcap.Ims.Default)
```

Puts a call on hold or resumes a call that has been put on hold. To select the call, use method RsCmwDau.Configure.Data. Control.Ims.Update.Call.Id.set.

param update_call_event HOLD | RESume

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.209 Id

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:CALL:ID
```

class Id

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

get(*ims=<Ims.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:CALL:ID
value: str = driver.configure.data.control.ims.update.call.id.get(ims = repcap.
↪Ims.Default)
```

Selects the call to be updated. To query a list of IDs, see method RsCmwDau.Configure.Data.Control.Ims.Release.Call.Id. set. All other UPDate commands affect the call selected via this command.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return idn: Call ID as string, selecting the call to be updated

set(*idn: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:CALL:ID
driver.configure.data.control.ims.update.call.id.set(idn = '1', ims = repcap.
↪Ims.Default)
```

Selects the call to be updated. To query a list of IDs, see method RsCmwDau.Configure.Data.Control.Ims.Release.Call.Id. set. All other UPDate commands affect the call selected via this command.

param idn Call ID as string, selecting the call to be updated

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.210 TypePy

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:CALL:TYPE
```

class TypePy

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.AvTypeA

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:CALL:TYPE
value: enums.AvTypeA = driver.configure.data.control.ims.update.call.typePy.
↪get(ims = repcap.Ims.Default)
```

Selects the new call type for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return call_type: AUDio | VIDeo

`set`(*call_type: RsCmwDau.enums.AvTypeA*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:CALL:TYPE
driver.configure.data.control.ims.update.call.typePy.set(call_type = enums.
↪AvTypeA.AUDio, ims = repcap.Ims.Default)
```

Selects the new call type for a call update.

param call_type AUDio | VIDeo

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.211 Evs

class Evs

Evs commands group definition. 14 total commands, 12 Sub-groups, 0 group commands

Cloning the Group

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

Subgroups

7.2.1.1.3.212 Codec<Codec>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.update.evs.codec.repcap_codec_get()
driver.configure.data.control.ims.update.evs.codec.repcap_codec_set(repcap.Codec.Ix1)
```

class Codec

Codec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Codec, default value after init: Codec.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.codec.clone()
```

Subgroups

7.2.1.1.3.213 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:CODec<Codec>:ENABle
```

class Enable

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

get(*ims=<Ims.Default: -1>*, *codec=<Codec.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:CODec<CodecIdx>:ENABle
value: bool = driver.configure.data.control.ims.update.evs.codec.enable.get(ims␣
↪= repcap.Ims.Default, codec = repcap.Codec.Default)
```

Enables or disables a codec rate for the AMR-WB IO mode of the EVS codec, for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

return codec_rate: OFF | ON OFF: codec rate not supported ON: codec rate supported

set(*codec_rate: bool*, *ims=<Ims.Default: -1>*, *codec=<Codec.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:CODec<CodecIdx>:ENABle
driver.configure.data.control.ims.update.evs.codec.enable.set(codec_rate =␣
↪False, ims = repcap.Ims.Default, codec = repcap.Codec.Default)
```

Enables or disables a codec rate for the AMR-WB IO mode of the EVS codec, for a call update.

param codec_rate OFF | ON OFF: codec rate not supported ON: codec rate supported

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

7.2.1.1.3.214 Common

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.common.clone()
```

Subgroups

7.2.1.1.3.215 Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.common.bitrate.clone()
```

Subgroups

7.2.1.1.3.216 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:COMMon:BITRate:RANGe
```

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

class RangeStruct
Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Lower end of the range, 5.9 kbit/s to 128 kbit/s
- Bitrate_Higher: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Upper end of the range, 5.9 kbit/s to 128 kbit/s

get(*ims*=<*Ims.Default: -1*>) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:COMMon:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.update.evs.common.
↪bitrate.range.get(ims = repcap.Ims.Default)
```

Selects the bit-rate range supported in the EVS primary mode, for a call update and common configuration.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.Update_.Evs_.Common_.Bitrate_.Range.Range.RangeStruct*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:COMMon:BITRate:RANGe
driver.configure.data.control.ims.update.evs.common.bitrate.range.set(value =␣
↪[PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default)
```

Selects the bit-rate range supported in the EVS primary mode, for a call update and common configuration.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.217 Receive

class Receive

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.receive.clone()
```

Subgroups

7.2.1.1.3.218 Bitrate

class Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.receive.bitrate.clone()
```

Subgroups

7.2.1.1.3.219 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:RECeive:BITRate:RANGe
```

class Range

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

class RangeStruct

Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Lower end of the range, 5.9 kbit/s to 128 kbit/s
- Bitrate_Higher: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Upper end of the range, 5.9 kbit/s to 128 kbit/s

get(*ims=<Ims.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:RECeive:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.update.evs.receive.
↪bitrate.range.get(ims = repcap.Ims.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the uplink (receive) direction, for a call update and separate configuration.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.Update_.Evs_.Receive_.Bitrate_.Range.Range.RangeStruct*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:RECeive:BITRate:RANGe
driver.configure.data.control.ims.update.evs.receive.bitrate.range.set(value =␣
↪[PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the uplink (receive) direction, for a call update and separate configuration.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.220 Bw

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:RECeive:BW
```

class Bw

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:RECeive:BW
value: enums.Bandwidth = driver.configure.data.control.ims.update.evs.receive.
↪bw.get(ims = repcap.Ims.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the uplink (receive) direction, for a call update and separate configuration.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return rx_bw: NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

set(*rx_bw: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:RECeive:BW
driver.configure.data.control.ims.update.evs.receive.bw.set(rx_bw = enums.
↪Bandwidth.FB, ims = repcap.Ims.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the uplink (receive) direction, for a call update and separate configuration.

param rx_bw NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.221 Send

class Send

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.send.clone()
```

Subgroups

7.2.1.1.3.222 Bw

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:SEND:BW
```

class Bw

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:SEND:BW
value: enums.Bandwidth = driver.configure.data.control.ims.update.evs.send.bw.
↪get(ims = repcap.Ims.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the downlink (send) direction, for a call update and separate configuration.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return send_bw: NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

set(*send_bw: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:SEND:BW
driver.configure.data.control.ims.update.evs.send.bw.set(send_bw = enums.
↪Bandwidth.FB, ims = repcap.Ims.Default)
```

Selects the codec bandwidths supported in the EVS primary mode in the downlink (send) direction, for a call update and separate configuration.

param send_bw NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.223 Bitrate

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.send.bitrate.clone()
```

Subgroups

7.2.1.1.3.224 Range

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:SEND:BITRate:RANGe
```

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

class RangeStruct
Structure for setting input parameters. Fields:

- Bitrate_Lower: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Lower end of the range, 5.9 kbit/s to 128 kbit/s
- Bitrate_Higher: enums.Bitrate: R59 | R72 | R80 | R96 | R132 | R164 | R244 | R320 | R480 | R640 | R960 | R1280 Upper end of the range, 5.9 kbit/s to 128 kbit/s

get(*ims=<Ims.Default: -1>*) → RangeStruct

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:SEND:BITRate:RANGe
value: RangeStruct = driver.configure.data.control.ims.update.evs.send.bitrate.
↪range.get(ims = repcap.Ims.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the downlink (send) direction, for a call update and separate configuration.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

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

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Control_.Ims_.Update_.Evs_.Send_.Bitrate_.Range.Range.RangeStruct*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:SEND:BITRate:RANGe
driver.configure.data.control.ims.update.evs.send.bitrate.range.set(value =␣
↪[PROPERTY_STRUCT_NAME](), ims = repcap.Ims.Default)
```

Selects the bit-rate range supported in the EVS primary mode in the downlink (send) direction, for a call update and separate configuration.

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

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.225 Synch

class Synch

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

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.evs.synch.clone()
```

Subgroups

7.2.1.1.3.226 Select

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:SYNCh:SELect
```

class Select

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.BwRange

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:SYNCh:SELect
value: enums.BwRange = driver.configure.data.control.ims.update.evs.synch.
↪select.get(ims = repcap.Ims.Default)
```

Selects a configuration mode for the bandwidth and bit-rate settings of the EVS primary mode, for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return bw_ranges: COMMon | SENDrx Common configuration or send/receive configured separately

set(*bw_ranges: RsCmwDau.enums.BwRange*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:SYNCh:SELect
driver.configure.data.control.ims.update.evs.synch.select.set(bw_ranges = enums.
↪BwRange.COMMon, ims = repcap.Ims.Default)
```

Selects a configuration mode for the bandwidth and bit-rate settings of the EVS primary mode, for a call update.

param bw_ranges COMMon | SENDrx Common configuration or send/receive configured separately

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.227 StartMode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:STARtmode
```

class StartMode

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.StartMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:STARtmode
value: enums.StartMode = driver.configure.data.control.ims.update.evs.startMode.
↪get(ims = repcap.Ims.Default)
```

Selects the start mode for the EVS codec, for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return start_mode: EPRimary | EAMRwbio EVS primary or EVS AMR-WB IO

set(*start_mode: RsCmwDau.enums.StartMode*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:STARtmode
driver.configure.data.control.ims.update.evs.startMode.set(start_mode = enums.
↪StartMode.EAMRwbio, ims = repcap.Ims.Default)
```

Selects the start mode for the EVS codec, for a call update.

param start_mode EPRimary | EAMRwbio EVS primary or EVS AMR-WB IO

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.228 ChawMode

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:CHAWmode
```

class ChawMode

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.ChawMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:CHAWmode
value: enums.ChawMode = driver.configure.data.control.ims.update.evs.chawMode.
↪get(ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'ch-aw-recv' for the EVS codec, for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return chaw_mode: DIS | NUSed | TWO | THRee | FIVE | SEVen | NP Disabled, not used, 2, 3, 5, 7, not present

set(*chaw_mode: RsCmwDau.enums.ChawMode*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:CHAWmode
driver.configure.data.control.ims.update.evs.chawMode.set(chaw_mode = enums.
↪ChawMode.DIS, ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'ch-aw-recv' for the EVS codec, for a call update.

param chaw_mode DIS | NUSed | TWO | THRee | FIVE | SEVen | NP Disabled, not used, 2, 3, 5, 7, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.229 Cmr

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:CMR
```

class Cmr

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.Cmr

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:CMR
value: enums.Cmr = driver.configure.data.control.ims.update.evs.cmr.get(ims =␣
↪repcap.Ims.Default)
```

Specifies the SDP parameter 'cmr' for the EVS codec, for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return cmr: DISable | ENABle | PRESent | NP Disable, enable, present all, not present

set(*cmr: RsCmwDau.enums.Cmr*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:CMR
driver.configure.data.control.ims.update.evs.cmr.set(cmr = enums.Cmr.DISable,␣
↪ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'cmr' for the EVS codec, for a call update.

param cmr DISable | ENABle | PRESent | NP Disable, enable, present all, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.230 DtxRecv

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:DTXRecv
```

class DtxRecv

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.DtxRecv

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:DTXRecv
value: enums.DtxRecv = driver.configure.data.control.ims.update.evs.dtxRecv.
↪get(ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'dtx-recv' for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return dtx_recv: DISable | ENABle | NP Disable, enable, not present

set(*dtx_recv: RsCmwDau.enums.DtxRecv*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:DTXRecv
driver.configure.data.control.ims.update.evs.dtxRecv.set(dtx_recv = enums.
↪DtxRecv.DISable, ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'dtx-recv' for a call update.

param dtx_recv DISable | ENABle | NP Disable, enable, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.231 Dtx

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:DTX
```

class Dtx

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.DtxRecv

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:DTX
value: enums.DtxRecv = driver.configure.data.control.ims.update.evs.dtx.get(ims␣
↪= repcap.Ims.Default)
```

Specifies the SDP parameter 'dtx' for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return dtx: DISable | ENABle | NP Disable, enable, not present

set(*dtx: RsCmwDau.enums.DtxRecv*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:DTX
driver.configure.data.control.ims.update.evs.dtx.set(dtx = enums.DtxRecv.
↪DISable, ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'dtx' for a call update.

param dtx DISable | ENABle | NP Disable, enable, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.232 HfOnly

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:HFONly
```

class HfOnly

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.HfOnly

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:HFONly
value: enums.HfOnly = driver.configure.data.control.ims.update.evs.hfOnly.
↪get(ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'hf-only' for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return hf: BOTH | HEADfull | NP Both, header-full only, not present

set(*hf: RsCmwDau.enums.HfOnly*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:HFONly
driver.configure.data.control.ims.update.evs.hfOnly.set(hf = enums.HfOnly.BOTH,␣
↪ims = repcap.Ims.Default)
```

Specifies the SDP parameter 'hf-only' for a call update.

param hf BOTH | HEADfull | NP Both, header-full only, not present

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.233 BwCommon

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:EVS:BWCommon
```

class BwCommon

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.Bandwidth

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:BWCommon
value: enums.Bandwidth = driver.configure.data.control.ims.update.evs.bwCommon.
↪get(ims = repcap.Ims.Default)
```

Selects the codec bandwidths supported in the EVS primary mode, for a call update and common configuration.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return bw_common: NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

set(*bw_common: RsCmwDau.enums.Bandwidth*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:EVS:BWCommon
driver.configure.data.control.ims.update.evs.bwCommon.set(bw_common = enums.
↪Bandwidth.FB, ims = repcap.Ims.Default)
```

Selects the codec bandwidths supported in the EVS primary mode, for a call update and common configuration.

param bw_common NB | WB | SWB | FB | NBWB | NBSWb | NBFB NB: narrowband only WB: wideband only SWB: super wideband only FB: fullband only NBWB: narrowband and wideband NBSWb: narrowband, wideband and super wideband NBFB: narrowband, wideband, super wideband and fullband

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.234 AdCodec

class AdCodec

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

Cloning the Group

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

Subgroups

7.2.1.1.3.235 TypePy

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:ADCodec:TYPE
```

class TypePy

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.CodecType

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:ADCodec:TYPE
value: enums.CodecType = driver.configure.data.control.ims.update.adCodec.
↪typePy.get(ims = repcap.Ims.Default)
```

Selects the new audio codec type for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return audio_codec: NARRowband | WIDeband | EVS AMR NB, AMR WB, EVS

set(*audio_codec: RsCmwDau.enums.CodecType*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:ADCodec:TYPE
driver.configure.data.control.ims.update.adCodec.typePy.set(audio_codec = enums.
↪CodecType.EVS, ims = repcap.Ims.Default)
```

Selects the new audio codec type for a call update.

param audio_codec NARRowband | WIDeband | EVS AMR NB, AMR WB, EVS

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.236 Amr

class Amr

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

Cloning the Group

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

Subgroups

7.2.1.1.3.237 Alignment

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:AMR:ALIGnment
```

class Alignment

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.AlignMode

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:AMR:ALIGnment
value: enums.AlignMode = driver.configure.data.control.ims.update.amr.alignment.
↪get(ims = repcap.Ims.Default)
```

Selects the new AMR voice codec alignment mode for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return amr_alignment: OCTetaligned | BANDwidtheff OCTetaligned: octet-aligned BANDwidtheff: bandwidth-efficient

set(*amr_alignment: RsCmwDau.enums.AlignMode*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:AMR:ALIGnment
driver.configure.data.control.ims.update.amr.alignment.set(amr_alignment =␣
↪enums.AlignMode.BANDwidtheff, ims = repcap.Ims.Default)
```

Selects the new AMR voice codec alignment mode for a call update.

param amr_alignment OCTetaligned | BANDwidtheff OCTetaligned: octet-aligned BANDwidtheff: bandwidth-efficient

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.238 Codec<Codec>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.update.amr.codec.repcap_codec_get()
driver.configure.data.control.ims.update.amr.codec.repcap_codec_set(repcap.Codec.Ix1)
```

class Codec

Codec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Codec, default value after init: Codec.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.update.amr.codec.clone()
```

Subgroups

7.2.1.1.3.239 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:AMR:CODec<Codec>:ENABle
```

class Enable

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

get(*ims=<Ims.Default: -1>*, *codec=<Codec.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:AMR:CODec<CodecIdx>:ENABle
value: bool = driver.configure.data.control.ims.update.amr.codec.enable.get(ims
↪= repcap.Ims.Default, codec = repcap.Codec.Default)
```

Enables or disables an AMR codec rate for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

return codec_rate: OFF | ON OFF: codec rate not supported ON: codec rate supported

set(*codec_rate: bool*, *ims=<Ims.Default: -1>*, *codec=<Codec.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:AMR:CODec<CodecIdx>:ENABle
driver.configure.data.control.ims.update.amr.codec.enable.set(codec_rate =
↪False, ims = repcap.Ims.Default, codec = repcap.Codec.Default)
```

Enables or disables an AMR codec rate for a call update.

param codec_rate OFF | ON OFF: codec rate not supported ON: codec rate supported

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

7.2.1.1.3.240 Video

class Video

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

Cloning the Group

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

Subgroups

7.2.1.1.3.241 Codec

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:VIDeo:CODec
```

class Codec

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

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.VideoCodec

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:VIDeo:CODec
value: enums.VideoCodec = driver.configure.data.control.ims.update.video.codec.
↪get(ims = repcap.Ims.Default)
```

Selects the video codec for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return video_codec: H263 | H264 H.263 or H.264 codec

set(*video_codec: RsCmwDau.enums.VideoCodec*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:VIDeo:CODec
driver.configure.data.control.ims.update.video.codec.set(video_codec = enums.
↪VideoCodec.H263, ims = repcap.Ims.Default)
```

Selects the video codec for a call update.

param video_codec H263 | H264 H.263 or H.264 codec

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.242 Attributes

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:VIDeo:ATTRibutes
```

class Attributes

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

get(*ims=<Ims.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:VIDeo:ATTRibutes
value: str = driver.configure.data.control.ims.update.video.attributes.get(ims␣
↪= repcap.Ims.Default)
```

Configures video codec attributes for a call update.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return video_attributes: Codec attributes as string

set(*video_attributes: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:VIDeo:ATTRibutes
driver.configure.data.control.ims.update.video.attributes.set(video_attributes␣
↪= '1', ims = repcap.Ims.Default)
```

Configures video codec attributes for a call update.

param video_attributes Codec attributes as string

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.243 Perform

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:UPDate:PERForm
```

class Perform

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

set(*ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:UPDate:PERForm
driver.configure.data.control.ims.update.perform.set(ims = repcap.Ims.Default)
```

Initiates an 'outband' call update. The settings configured via the UPDate commands are applied to the call selected via method RsCmwDau.Configure.Data.Control.Ims.Update.Call.Id.set.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

set_with_opc(*ims=<Ims.Default: -1>*) → None

7.2.1.1.3.244 Release

class Release

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

Cloning the Group

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

Subgroups

7.2.1.1.3.245 Call

class Call

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

Cloning the Group

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

Subgroups

7.2.1.1.3.246 Id

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS<Ims>:RELease:CALL:ID
```

class Id

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

get(*ims=<Ims.Default: -1>*) → List[str]

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:RELease:CALL:ID
value: List[str] = driver.configure.data.control.ims.release.call.id.get(ims =␣
↪repcap.Ims.Default)
```

Queries a list of call IDs or releases a call selected via its ID.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return ids: Comma-separated list of ID strings, one string per established call

set(*idn: str*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS<Suffix>:RELease:CALL:ID
driver.configure.data.control.ims.release.call.id.set(idn = '1', ims = repcap.
↪Ims.Default)
```

Queries a list of call IDs or releases a call selected via its ID.

param idn ID as string, selecting the call to be released

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.2.1.1.3.247 Sms

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:SMS:TYPE
CONFigure:DATA:CONTrol:IMS:SMS:TEXT
CONFigure:DATA:CONTrol:IMS:SMS:SEND
```

class Sms

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

get_text() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:SMS:TEXT
value: str = driver.configure.data.control.ims.sms.get_text()
```

No command help available

return sms_text: No help available

get_type_py() → RsCmwDau.enums.SmsTypeB

```
# SCPI: CONFigure:DATA:CONTrol:IMS:SMS:TYPE
value: enums.SmsTypeB = driver.configure.data.control.ims.sms.get_type_py()
```

No command help available

return sms_type: No help available

send(*sms_text: Optional[str] = None*, *sms_type: Optional[RsCmwDau.enums.SmsTypeB] = None*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:SMS:SEND
driver.configure.data.control.ims.sms.send(sms_text = '1', sms_type = enums.
↪SmsTypeB.TGP2)
```

No command help available

param sms_text No help available

param sms_type No help available

set_text(*sms_text: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:SMS:TEXT
driver.configure.data.control.ims.sms.set_text(sms_text = '1')
```

No command help available

param sms_text No help available

set_type_py(*sms_type: RsCmwDau.enums.SmsTypeB*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:SMS:TYPE
driver.configure.data.control.ims.sms.set_type_py(sms_type = enums.SmsTypeB.
↪TGP2)
```

No command help available

param sms_type No help available

7.2.1.1.3.248 Voice

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:VOICe:AUDiorouting
CONFigure:DATA:CONTrol:IMS:VOICe:TYPE
CONFigure:DATA:CONTrol:IMS:VOICe:AMRType
CONFigure:DATA:CONTrol:IMS:VOICe:VCODec
CONFigure:DATA:CONTrol:IMS:VOICe:LOOPback
CONFigure:DATA:CONTrol:IMS:VOICe:PRECondition
```

class Voice

Voice commands group definition. 11 total commands, 3 Sub-groups, 6 group commands

get_amr_type() → RsCmwDau.enums.AmrType

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:AMRType
value: enums.AmrType = driver.configure.data.control.ims.voice.get_amr_type()
```

No command help available

return amr_type: No help available

get_audio_routing() → RsCmwDau.enums.AudioRouting

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:AUDiorouting
value: enums.AudioRouting = driver.configure.data.control.ims.voice.get_audio_
↪routing()
```

No command help available

return audio_routing: No help available

get_loopback() → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:LOOPback
value: bool = driver.configure.data.control.ims.voice.get_loopback()
```

No command help available

return use_loopback: No help available

get_precondition() → RsCmwDau.enums.VoicePrecondition

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:PRECondition
value: enums.VoicePrecondition = driver.configure.data.control.ims.voice.get_
↪precondition()
```

No command help available

return voice_precon: No help available

get_type_py() → RsCmwDau.enums.AvTypeA

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:TYPE
value: enums.AvTypeA = driver.configure.data.control.ims.voice.get_type_py()
```

No command help available

return call_type: No help available

get_vcodec() → RsCmwDau.enums.VideoCodec

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:VCODec
value: enums.VideoCodec = driver.configure.data.control.ims.voice.get_vcodec()
```

No command help available

return video_codec: No help available

set_amr_type(*amr_type: RsCmwDau.enums.AmrType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:AMRType
driver.configure.data.control.ims.voice.set_amr_type(amr_type = enums.AmrType.
↪NARRowband)
```

No command help available

param amr_type No help available

set_audio_routing(*audio_routing: RsCmwDau.enums.AudioRouting*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:AUDiorouting
driver.configure.data.control.ims.voice.set_audio_routing(audio_routing = enums.
↪AudioRouting.AUDioboard)
```

No command help available

param audio_routing No help available

set_loopback(*use_loopback: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:LOOPback
driver.configure.data.control.ims.voice.set_loopback(use_loopback = False)
```

No command help available

param use_loopback No help available

set_precondition(*voice_precon: RsCmwDau.enums.VoicePrecondition*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:PRECondition
driver.configure.data.control.ims.voice.set_precondition(voice_precon = enums.
↪VoicePrecondition.SIMPle)
```

No command help available

param voice_precon No help available

set_type_py(*call_type: RsCmwDau.enums.AvTypeA*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:TYPE
driver.configure.data.control.ims.voice.set_type_py(call_type = enums.AvTypeA.
↪AUDio)
```

No command help available

param call_type No help available

set_vcodec(*video_codec: RsCmwDau.enums.VideoCodec*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:VCODec
driver.configure.data.control.ims.voice.set_vcodec(video_codec = enums.
↪VideoCodec.H263)
```

No command help available

param video_codec No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.voice.clone()
```

Subgroups

7.2.1.1.3.249 Codec<Codec>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.ims.voice.codec.repcap_codec_get()
driver.configure.data.control.ims.voice.codec.repcap_codec_set(repcap.Codec.Ix1)
```

class Codec

Codec commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Codec, default value after init: Codec.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.voice.codec.clone()
```

Subgroups

7.2.1.1.3.250 Enable

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:VOICe:CODec<Codec>:ENABle
```

class Enable

Enable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*codec*=<*Codec.Default: -1*>) → bool

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:CODec<codecIdx>:ENABle
value: bool = driver.configure.data.control.ims.voice.codec.enable.get(codec =␣
↪repcap.Codec.Default)
```

No command help available

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

return codec_use: No help available

set(*codec_use: bool*, *codec*=<*Codec.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:CODec<codecIdx>:ENABle
driver.configure.data.control.ims.voice.codec.enable.set(codec_use = False,␣
↪codec = repcap.Codec.Default)
```

No command help available

param codec_use No help available

param codec optional repeated capability selector. Default value: Ix1 (settable in the interface 'Codec')

7.2.1.1.3.251 MendPoint

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:VOICe:MENDpoint:PORT
CONFigure:DATA:CONTrol:IMS:VOICe:MENDpoint:IPADdress
```

class MendPoint

MendPoint commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ip_address() → str

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:MENDpoint:IPADdress
value: str = driver.configure.data.control.ims.voice.mendPoint.get_ip_address()
```

No command help available

return ip_address: No help available

get_port() → int

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:MENDpoint:PORT
value: int = driver.configure.data.control.ims.voice.mendPoint.get_port()
```

No command help available

return port: No help available

set_ip_address(*ip_address: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:MENDpoint:IPADdress
driver.configure.data.control.ims.voice.mendPoint.set_ip_address(ip_address = '1
↪')
```

No command help available

param ip_address No help available

set_port(*port: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:MENDpoint:PORT
driver.configure.data.control.ims.voice.mendPoint.set_port(port = 1)
```

No command help available

param port No help available

7.2.1.1.3.252 Call

class Call
Call commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ims.voice.call.clone()
```

Subgroups

7.2.1.1.3.253 Establish

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:VOICe:CALL:ESTablish
```

class Establish
Establish commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:CALL:ESTablish
driver.configure.data.control.ims.voice.call.establish.set()
```

No command help available

set_with_opc() → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:CALL:ESTablish
driver.configure.data.control.ims.voice.call.establish.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.1.3.254 Disconnect

SCPI Commands

```
CONFigure:DATA:CONTrol:IMS:VOICe:CALL:DISConnect
```

class Disconnect
Disconnect commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:CALL:DISConnect
driver.configure.data.control.ims.voice.call.disconnect.set()
```

No command help available

set_with_opc() → None

```
# SCPI: CONFigure:DATA:CONTrol:IMS:VOICe:CALL:DISConnect
driver.configure.data.control.ims.voice.call.disconnect.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.1.4 Supl

SCPI Commands

```
CONFigure:DATA:CONTrol:SUPL:TRANsmit
```

class Supl

Supl commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set_transmit(*message: bytes*) → None

```
# SCPI: CONFigure:DATA:CONTrol:SUPL:TRANsmit
driver.configure.data.control.supl.set_transmit(message = b'ABCDEFGH')
```

No command help available

param message No help available

7.2.1.1.5 IpvSix

class IpvSix

IpvSix commands group definition. 10 total commands, 6 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ipvSix.clone()
```

Subgroups

7.2.1.1.5.1 Prefixes

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:PREFixes:POOL
```

class Prefixes

Prefixes commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_pool() → bool

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:PREFixes:POOL
value: bool = driver.configure.data.control.ipvSix.prefixes.get_pool()
```

Enables or disables prefix delegation for automatic IPv6 configuration.

return prefix_pool: OFF | ON

set_pool(*prefix_pool: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:PREFixes:POOL
driver.configure.data.control.ipvSix.prefixes.set_pool(prefix_pool = False)
```

Enables or disables prefix delegation for automatic IPv6 configuration.

param prefix_pool OFF | ON

7.2.1.1.5.2 Address

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:ADDRess:TYPE
```

class Address

Address commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_type_py() → RsCmwDau.enums.AddressModeB

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:ADDRess:TYPE
value: enums.AddressModeB = driver.configure.data.control.ipvSix.address.get_
↪type_py()
```

Selects the method to be used for IPv6 DAU address configuration.

return address_type: AUTO | STATic | ACONf AUTO: predefined automatic configuration (standalone setup) STATic: static IP configuration ACONf: dynamic autoconfiguration

set_type_py(*address_type: RsCmwDau.enums.AddressModeB*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:ADDRess:TYPE
driver.configure.data.control.ipvSix.address.set_type_py(address_type = enums.
↪AddressModeB.ACONf)
```

Selects the method to be used for IPv6 DAU address configuration.

param address_type AUTO | STATic | ACONf AUTO: predefined automatic configuration (standalone setup) STATic: static IP configuration ACONf: dynamic autoconfiguration

7.2.1.1.5.3 Static

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:STATic:ADDRess
CONFigure:DATA:CONTrol:IPVSix:STATic:DROuter
```

class Static

Static commands group definition. 4 total commands, 1 Sub-groups, 2 group commands

get_address() → str

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:STATic:ADDRess
value: str = driver.configure.data.control.ipvSix.static.get_address()
```

Sets the IP address of the DAU to be used for static IPv6 configuration.

return ip_address: IPv6 address as string

get_drouter() → str

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:STATic:DROuter
value: str = driver.configure.data.control.ipvSix.static.get_drouter()
```

Sets the IP address of the external default router to be used for static IPv6 configuration.

return router: IPv6 address as string

set_address(*ip_address: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:STATic:ADDRess
driver.configure.data.control.ipvSix.static.set_address(ip_address = '1')
```

Sets the IP address of the DAU to be used for static IPv6 configuration.

param ip_address IPv6 address as string

set_drouter(*router: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:STATic:DROuter
driver.configure.data.control.ipvSix.static.set_drouter(router = '1')
```

Sets the IP address of the external default router to be used for static IPv6 configuration.

param router IPv6 address as string

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ipvSix.static.clone()
```

Subgroups

7.2.1.1.5.4 Prefixes

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:STATic:PREFixes:ADD
CONFigure:DATA:CONTrol:IPVSix:STATic:PREFixes:DELete
```

class Prefixes

Prefixes commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

delete(*prefix: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:STATic:PREFixes:DELete
driver.configure.data.control.ipvSix.static.prefixes.delete(prefix = 1)
```

Deletes an entry from the IPv6 prefix pool for DUTs, for static IPv6 configuration.

param prefix Entry to be deleted, either identified via its index number or its prefix string Range: 0 to total number of entries - 1 | 'prefix'

set_add(*prefix: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:STATic:PREFixes:ADD
driver.configure.data.control.ipvSix.static.prefixes.set_add(prefix = '1')
```

Adds a new prefix to the IPv6 prefix pool for DUTs, for static IPv6 configuration.

param prefix String, e.g. 'fcb1:abab:cdcd:efe0::/64'

7.2.1.1.5.5 Mobile

class Mobile

Mobile commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ipvSix.mobile.clone()
```

Subgroups

7.2.1.1.5.6 Prefix

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:MOBile:PREFix:TYPE
```

class Prefix

Prefix commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_type_py() → RsCmwDau.enums.PrefixType

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:MOBile:PREFix:TYPE
value: enums.PrefixType = driver.configure.data.control.ipvSix.mobile.prefix.
↪get_type_py()
```

Selects the method to be used to define the IPv6 prefix pool for DUTs. This setting is only relevant for test setups with connected external network. It is ignored for a standalone test setup (AUTO set via method RsCmwDau.Configure.Data. Control.IpvSix.Address.typePy) .

return prefix_type: STATic | DHCP STATic: static IP configuration DHCP: DHCP prefix delegation

set_type_py(*prefix_type: RsCmwDau.enums.PrefixType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:MOBile:PREFix:TYPE
driver.configure.data.control.ipvSix.mobile.prefix.set_type_py(prefix_type =␣
↪enums.PrefixType.DHCP)
```

Selects the method to be used to define the IPv6 prefix pool for DUTs. This setting is only relevant for test setups with connected external network. It is ignored for a standalone test setup (AUTO set via method RsCmwDau.Configure.Data. Control.IpvSix.Address.typePy) .

param prefix_type STATic | DHCP STATic: static IP configuration DHCP: DHCP prefix delegation

7.2.1.1.5.7 Routing

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:ROUTing:TYPE
```

class Routing

Routing commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_type_py() → RsCmwDau.enums.RoutingType

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:ROUTing:TYPE
value: enums.RoutingType = driver.configure.data.control.ipvSix.routing.get_
↪type_py()
```

Selects the mechanism to be used for IPv6 route configuration. The routes are only relevant for mobile-originating packets with destination addresses that do not belong to the subnet of the DAU and that are not reachable via the default router.

return routing_type: MANual In the current software version, the value is fixed. MANual: manually configured routes

set_type_py(*routing_type: RsCmwDau.enums.RoutingType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:ROUTing:TYPE
driver.configure.data.control.ipvSix.routing.set_type_py(routing_type = enums.
↪RoutingType.MANual)
```

Selects the mechanism to be used for IPv6 route configuration. The routes are only relevant for mobile-originating packets with destination addresses that do not belong to the subnet of the DAU and that are not reachable via the default router.

param routing_type MANual In the current software version, the value is fixed. MANual: manually configured routes

7.2.1.1.5.8 Manual

class Manual

Manual commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ipvSix.manual.clone()
```

Subgroups

7.2.1.1.5.9 Routing

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:MANual:ROUTing:DELete
```

class Routing

Routing commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

delete(*prefix: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:MANual:ROUTing:DELete
driver.configure.data.control.ipvSix.manual.routing.delete(prefix = 1)
```

Deletes an entry from the pool of manual routes for IPv6.

param prefix Entry to be deleted, either identified via its index number or its prefix string Range: 0 to total number of entries - 1 | 'prefix'

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ipvSix.manual.routing.clone()
```

Subgroups

7.2.1.1.5.10 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVSix:MANual:ROUTing:ADD
```

class Add

Add commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set(*prefix: str*, *router: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVSix:MANual:ROUTing:ADD
driver.configure.data.control.ipvSix.manual.routing.add.set(prefix = '1',␣
↪router = '1')
```

Adds a route to the pool of manual routes for IPv6. If the destination address of a packet matches the <Prefix>, it is routed to the <Router>.

param prefix String, e.g. 'fcb1:abab:cdcd:efe0::/64', 64-bit prefixes and shorter prefixes allowed

param router Router address as string, e.g. 'fcb1:abcd:17c5:efe0::1'

7.2.1.1.6 Advanced

SCPI Commands

```
CONFigure:DATA:CONTrol:ADVanced:IPBuffering
```

class Advanced

Advanced commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_ip_buffering() → bool

```
# SCPI: CONFigure:DATA:CONTrol:ADVanced:IPBuffering
value: bool = driver.configure.data.control.advanced.get_ip_buffering()
```

No command help available

return buffering_cnfg: No help available

set_ip_buffering(*buffering_cnfg: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:ADVanced:IPBuffering
driver.configure.data.control.advanced.set_ip_buffering(buffering_cnfg = False)
```

No command help available

param buffering_cnfg No help available

7.2.1.1.7 IpvFour

class IpvFour

IpvFour commands group definition. 6 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ipvFour.clone()
```

Subgroups

7.2.1.1.7.1 Address

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVFour:ADDRess:TYPE
```

class Address

Address commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_type_py() → RsCmwDau.enums.AddressModeA

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:ADDRess:TYPE
value: enums.AddressModeA = driver.configure.data.control.ipvFour.address.get_
↪type_py()
```

Selects the type of the IPv4 configuration.

return address_type: AUTomatic | STATic | DHCPv4 AUTomatic: predefined internal IP configuration STATic: user-defined static IP configuration defined via the commands CONFigure:DATA:CONTrol:IPVFour:STATic:... DHCPv4: the IPv4 address is obtained from a DHCP server in the company LAN

set_type_py(*address_type: RsCmwDau.enums.AddressModeA*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:ADDRess:TYPE
driver.configure.data.control.ipvFour.address.set_type_py(address_type = enums.
↪AddressModeA.AUTomatic)
```

Selects the type of the IPv4 configuration.

param address_type AUTomatic | STATic | DHCPv4 AUTomatic: predefined internal IP configuration STATic: user-defined static IP configuration defined via the commands CONFigure:DATA:CONTrol:IPVFour:STATic:... DHCPv4: the IPv4 address is obtained from a DHCP server in the company LAN

7.2.1.1.7.2 Static

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVFour:STATic:GIP
CONFigure:DATA:CONTrol:IPVFour:STATic:IPADdress
CONFigure:DATA:CONTrol:IPVFour:STATic:SMASk
```

class Static

Static commands group definition. 5 total commands, 1 Sub-groups, 3 group commands

get_gip() → str

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:GIP
value: str = driver.configure.data.control.ipvFour.static.get_gip()
```

Defines the address of an external gateway to be used for static IPv4 configuration.

return gateway_ip: IP address as string Range: '0.0.0.0' to '255.255.255.255'

get_ip_address() → str

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:IPADdress
value: str = driver.configure.data.control.ipvFour.static.get_ip_address()
```

Sets the IP address of the DAU to be used for static IPv4 configuration.

return ip_address: IP address as string Range: '0.0.0.0' to '255.255.255.255'

get_smask() → str

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:SMASk
value: str = driver.configure.data.control.ipvFour.static.get_smask()
```

Defines the subnet mask for static IPv4 configuration.

return subnet_mask: Subnet mask as string Range: '0.0.0.0' to '255.255.255.255'

set_gip(*gateway_ip: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:GIP
driver.configure.data.control.ipvFour.static.set_gip(gateway_ip = '1')
```

Defines the address of an external gateway to be used for static IPv4 configuration.

param gateway_ip IP address as string Range: '0.0.0.0' to '255.255.255.255'

set_ip_address(*ip_address: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:IPADdress
driver.configure.data.control.ipvFour.static.set_ip_address(ip_address = '1')
```

Sets the IP address of the DAU to be used for static IPv4 configuration.

param ip_address IP address as string Range: '0.0.0.0' to '255.255.255.255'

set_smask(*subnet_mask: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:SMASk
driver.configure.data.control.ipvFour.static.set_smask(subnet_mask = '1')
```

Defines the subnet mask for static IPv4 configuration.

param subnet_mask Subnet mask as string Range: '0.0.0.0' to '255.255.255.255'

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ipvFour.static.clone()
```

Subgroups

7.2.1.1.7.3 Addresses

SCPI Commands

```
CONFigure:DATA:CONTrol:IPVFour:STATic:ADDResses:ADD
CONFigure:DATA:CONTrol:IPVFour:STATic:ADDResses:DELete
```

class Addresses

Addresses commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

delete(*ip_address: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:ADDResses:DELete
driver.configure.data.control.ipvFour.static.addresses.delete(ip_address = 1)
```

Deletes an address from the IPv4 address pool for DUTs, for static IPv4 configuration.

param ip_address IP address to be deleted, either identified via its index number or as string Range: 0 to total number of entries - 1 | '0.0.0.0' to '255.255.255.255'

set_add(*ip_address: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:IPVFour:STATic:ADDResses:ADD
driver.configure.data.control.ipvFour.static.addresses.set_add(ip_address = '1')
```

Adds an IP address to the IPv4 address pool for DUTs, for static IPv4 configuration.

param ip_address IP address as string Range: '0.0.0.0' to '255.255.255.255'

7.2.1.1.8 Dns

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:RESallquery
```

class Dns

Dns commands group definition. 18 total commands, 6 Sub-groups, 1 group commands

get_res_all_query() → bool

```
# SCPI: CONFigure:DATA:CONTrol:DNS:RESallquery
value: bool = driver.configure.data.control.dns.get_res_all_query()
```

Configures the response of the internal DNS server, if no matching database entry is found for a DNS query of type A or AAAA.

return dns_resolve_all: OFF | ON OFF: Return no IP address ON: Return the DAU IP address

set_res_all_query(*dns_resolve_all: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:RESallquery
driver.configure.data.control.dns.set_res_all_query(dns_resolve_all = False)
```

Configures the response of the internal DNS server, if no matching database entry is found for a DNS query of type A or AAAA.

param dns_resolve_all OFF | ON OFF: Return no IP address ON: Return the DAU IP address

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.dns.clone()
```

Subgroups

7.2.1.1.8.1 Primary

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:PRIMary:STYPe
```

class Primary

Primary commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_stype() → RsCmwDau.enums.ServerType

```
# SCPI: CONFigure:DATA:CONTrol:DNS:PRIMary:STYPe
value: enums.ServerType = driver.configure.data.control.dns.primary.get_stype()
```

Select the primary and secondary DNS server type.

return stype: NONE | INTernal | IAForeign | FOReign NONE: no DNS server address sent to the DUT INTernal: use local DNS server IAForeign: use local DNS server, if no entry found then foreign DNS server FOReign: use foreign DNS server

set_stype(*stype: RsCmwDau.enums.ServerType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:PRIMary:STYPe
driver.configure.data.control.dns.primary.set_stype(stype = enums.ServerType.
↪FOReign)
```

Select the primary and secondary DNS server type.

param stype NONE | INTernal | IAForeign | FOReign NONE: no DNS server address sent to the DUT INTernal: use local DNS server IAForeign: use local DNS server, if no entry found then foreign DNS server FOReign: use foreign DNS server

7.2.1.1.8.2 Secondary

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:SECondary:STYPe
```

class Secondary

Secondary commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_stype() → RsCmwDau.enums.ServerType

```
# SCPI: CONFigure:DATA:CONTrol:DNS:SECondary:STYPe
value: enums.ServerType = driver.configure.data.control.dns.secondary.get_
↪stype()
```

Select the primary and secondary DNS server type.

return stype: NONE | INTernal | IAForeign | FOReign NONE: no DNS server address sent to the DUT INTernal: use local DNS server IAForeign: use local DNS server, if no entry found then foreign DNS server FOReign: use foreign DNS server

set_stype(*stype: RsCmwDau.enums.ServerType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:SECondary:STYPe
driver.configure.data.control.dns.secondary.set_stype(stype = enums.ServerType.
↪FOReign)
```

Select the primary and secondary DNS server type.

param stype NONE | INTernal | IAForeign | FOReign NONE: no DNS server address sent to the DUT INTernal: use local DNS server IAForeign: use local DNS server, if no entry found then foreign DNS server FOReign: use foreign DNS server

7.2.1.1.8.3 Foreign

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:FOReign:UDHCp
```

class Foreign

Foreign commands group definition. 9 total commands, 2 Sub-groups, 1 group commands

class UdhcpStruct

Structure for reading output parameters. Fields:

- Prim_Ip_4: bool: OFF | ON DHCPv4 address, primary DNS server
- Prim_Ip_6: bool: OFF | ON DHCPv6 address, primary DNS server
- Sec_Ip_4: bool: OFF | ON DHCPv4 address, secondary DNS server
- Sec_Ip_6: bool: OFF | ON DHCPv6 address, secondary DNS server

get_udhcp() → UdhcpStruct

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:UDHCp
value: UdhcpStruct = driver.configure.data.control.dns.foreign.get_udhcp()
```

Specifies whether an IP address received via DHCPv4 / DHCPv6 is used (if available) instead of the IPv4 / IPv6 address configured statically for the foreign DNS server.

return structure: for return value, see the help for UdhcpStruct structure arguments.

set_udhcp(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Dns_.Foreign.Foreign.UdhcpStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:UDHCp
driver.configure.data.control.dns.foreign.set_udhcp(value = UdhcpStruct())
```

Specifies whether an IP address received via DHCPv4 / DHCPv6 is used (if available) instead of the IPv4 / IPv6 address configured statically for the foreign DNS server.

param value see the help for UdhcpStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.dns.foreign.clone()
```

Subgroups

7.2.1.1.8.4 IpvFour

class IpvFour

IpvFour commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.dns.foreign.ipvFour.clone()
```

Subgroups

7.2.1.1.8.5 Primary

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:PRIMary:ADDRess
CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:PRIMary:UDHCp
```

class Primary

Primary commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_address() → str

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:PRIMary:ADDRess
value: str = driver.configure.data.control.dns.foreign.ipvFour.primary.get_
↪address()
```

Specifies the IPv4 address of the foreign primary DNS server.

return fdns_prim_ip_4: IPv4 address as string Range: '0.0.0.0' to '255.255.255.255'

get_udhcp() → bool

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:PRIMary:UDHCp
value: bool = driver.configure.data.control.dns.foreign.ipvFour.primary.get_
↪udhcp()
```

No command help available

return use_dhcpip_4: No help available

set_address(*fdns_prim_ip_4: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:PRIMary:ADDRess
driver.configure.data.control.dns.foreign.ipvFour.primary.set_address(fdns_prim_
↪ip_4 = '1')
```

Specifies the IPv4 address of the foreign primary DNS server.

param fdns_prim_ip_4 IPv4 address as string Range: '0.0.0.0' to '255.255.255.255'

set_udhcp(*use_dhcpip_4: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:PRIMary:UDHCp
driver.configure.data.control.dns.foreign.ipvFour.primary.set_udhcp(use_dhcpip_
↪4 = False)
```

No command help available

param use_dhcpip_4 No help available

7.2.1.1.8.6 Secondary

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:SECondary:ADDRess
CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:SECondary:UDHCp
```

class Secondary

Secondary commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_address() → str

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:SECondary:ADDRess
value: str = driver.configure.data.control.dns.foreign.ipvFour.secondary.get_
↪address()
```

Specifies the IPv4 address of the foreign secondary DNS server.

return fdns_sec_ip_4: IPv4 address as string Range: '0.0.0.0' to '255.255.255.255'

get_udhcp() → bool

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:SECondary:UDHCp
value: bool = driver.configure.data.control.dns.foreign.ipvFour.secondary.get_
↪udhcp()
```

No command help available

return use_dhcpip_4: No help available

set_address(*fdns_sec_ip_4: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:SECondary:ADDRess
driver.configure.data.control.dns.foreign.ipvFour.secondary.set_address(fdns_
↪sec_ip_4 = '1')
```

Specifies the IPv4 address of the foreign secondary DNS server.

param fdns_sec_ip_4 IPv4 address as string Range: '0.0.0.0' to '255.255.255.255'

set_udhcp(*use_dhcpip_4: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVFour:SECondary:UDHCp
driver.configure.data.control.dns.foreign.ipvFour.secondary.set_udhcp(use_
↪dhcpip_4 = False)
```

No command help available

param use_dhcpip_4 No help available

7.2.1.1.8.7 IpvSix

class IpvSix

IpvSix commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.dns.foreign.ipvSix.clone()
```

Subgroups

7.2.1.1.8.8 Primary

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:PRIMary:UDHCp
CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:PRIMary:ADDRess
```

class Primary

Primary commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_address() → str

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:PRIMary:ADDRess
value: str = driver.configure.data.control.dns.foreign.ipvSix.primary.get_
↪address()
```

Specifies the IPv6 address of the foreign primary DNS server.

return fdns_prim_ip_6: IPv6 address as string

get_udhcp() → bool

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:PRIMary:UDHCp
value: bool = driver.configure.data.control.dns.foreign.ipvSix.primary.get_
↪udhcp()
```

No command help available

return use_dhcpip_6: No help available

set_address(*fdns_prim_ip_6: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:PRIMary:ADDRess
driver.configure.data.control.dns.foreign.ipvSix.primary.set_address(fdns_prim_
↪ip_6 = '1')
```

Specifies the IPv6 address of the foreign primary DNS server.

param fdns_prim_ip_6 IPv6 address as string

set_udhcp(*use_dhcpip_6: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:PRIMary:UDHCp
driver.configure.data.control.dns.foreign.ipvSix.primary.set_udhcp(use_dhcpip_6␣
↪= False)
```

No command help available

param use_dhcpip_6 No help available

7.2.1.1.8.9 Secondary

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:SECondary:UDHCp
CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:SECondary:ADDRess
```

class Secondary

Secondary commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_address() → str

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:SECondary:ADDRess
value: str = driver.configure.data.control.dns.foreign.ipvSix.secondary.get_
↪address()
```

Specifies the IPv6 address of the foreign secondary DNS server.

return fdns_sec_ip_6: IPv6 address as string

get_udhcp() → bool

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:SECondary:UDHCp
value: bool = driver.configure.data.control.dns.foreign.ipvSix.secondary.get_
↪udhcp()
```

No command help available

return use_dhcpip_6: No help available

set_address(*fdns_sec_ip_6: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:SECondary:ADDRess
driver.configure.data.control.dns.foreign.ipvSix.secondary.set_address(fdns_sec_
↪ip_6 = '1')
```

Specifies the IPv6 address of the foreign secondary DNS server.

param fdns_sec_ip_6 IPv6 address as string

set_udhcp(*use_dhcpip_6: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:FOReign:IPVSix:SECondary:UDHCp
driver.configure.data.control.dns.foreign.ipvSix.secondary.set_udhcp(use_dhcpip_
↪6 = False)
```

No command help available

param use_dhcpip_6 No help available

7.2.1.1.8.10 Local

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:LOCal:DELete
```

class Local

Local commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

delete(*url_or_ip: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:LOCal:DELete
driver.configure.data.control.dns.local.delete(url_or_ip = '1')
```

Deletes an entry from the database of the local DNS server for type A or type AAAA DNS queries. Each entry consists of two strings, one specifying a domain and the other indicating the assigned IP address. Enter one of these strings to select the entry to be deleted.

param url_or_ip String selecting the entry to be deleted

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.dns.local.clone()
```

Subgroups

7.2.1.1.8.11 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:LOCal:ADD
```

class Add

Add commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set(*url: str*, *ip: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:LOCal:ADD
driver.configure.data.control.dns.local.add.set(url = '1', ip = '1')
```

Adds an entry to the database of the local DNS server for type A or type AAAA DNS queries. Each entry consists of two strings, one specifying a domain and the other indicating the assigned IP address.

param url String specifying the URL of a domain, e.g. 'www.example.com'

param ip Assigned IPv4 address or IPv6 address as string, e.g. '192.168.168.170' or 'fcb1:abab:1::1'

7.2.1.1.8.12 Aservices

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:ASERvices:DELete
```

class Aservices

Aservices commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

delete(*name: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:ASERvices:DELete
driver.configure.data.control.dns.aservices.delete(name = '1')
```

Deletes an entry from the database of the local DNS server for type SRV DNS queries.

param name String specifying the service name of the entry to be deleted

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.dns.aservices.clone()
```

Subgroups

7.2.1.1.8.13 Add

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:ASERvices:ADD
```

class Add

Add commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set(*name: str*, *url: str*, *protocol: RsCmwDau.enums.Protocol*, *port: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:ASERvices:ADD
driver.configure.data.control.dns.aservices.add.set(name = '1', url = '1',␣
↪protocol = enums.Protocol.TCP, port = 1)
```

Adds an entry to the database of the local DNS server for type SRV DNS queries.

param name String specifying the service name, e.g. 'pcscf'

param url String specifying the URL of the domain, e.g. 'www.example.com'

param protocol UDP | TCP

param port Range: 0 to 65654

7.2.1.1.8.14 Test

SCPI Commands

```
CONFigure:DATA:CONTrol:DNS:TEST:DOMain
CONFigure:DATA:CONTrol:DNS:TEST:STARt
```

class Test

Test commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_domain() → str

```
# SCPI: CONFigure:DATA:CONTrol:DNS:TEST:DOMain
value: str = driver.configure.data.control.dns.test.get_domain()
```

Specifies the domain to be resolved during a test of the foreign DNS server.

return domain: String specifying the URL of the domain

set_domain(*domain: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:TEST:DOMain
driver.configure.data.control.dns.test.set_domain(domain = '1')
```

Specifies the domain to be resolved during a test of the foreign DNS server.

param domain String specifying the URL of the domain

start() → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:TEST:STARt
driver.configure.data.control.dns.test.start()
```

Starts a test of the foreign DNS server.

start_with_opc() → None

```
# SCPI: CONFigure:DATA:CONTrol:DNS:TEST:STARt
driver.configure.data.control.dns.test.start_with_opc()
```

Starts a test of the foreign DNS server.

Same as start, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.1.9 Ftp

SCPI Commands

```
CONFigure:DATA:CONTrol:FTP:STYPe
CONFigure:DATA:CONTrol:FTP:ENConnection
CONFigure:DATA:CONTrol:FTP:AUSer
CONFigure:DATA:CONTrol:FTP:DUPLoad
CONFigure:DATA:CONTrol:FTP:IPVSix
```

class Ftp

Ftp commands group definition. 7 total commands, 1 Sub-groups, 5 group commands

get_auser() → bool

```
# SCPI: CONFigure:DATA:CONTrol:FTP:AUSer
value: bool = driver.configure.data.control.ftp.get_auser()
```

Specifies whether access to the FTP server is allowed for anonymous users.

return anonymous: OFF | ON OFF: not allowed ON: allowed

get_dupload() → bool

```
# SCPI: CONFigure:DATA:CONTrol:FTP:DUPLoad
value: bool = driver.configure.data.control.ftp.get_dupload()
```

Specifies whether data upload to the FTP server is allowed for anonymous users.

return data_upload: OFF | ON OFF: not allowed ON: allowed

get_en_connection() → bool

```
# SCPI: CONFigure:DATA:CONTrol:FTP:ENConnection
value: bool = driver.configure.data.control.ftp.get_en_connection()
```

Specifies whether access to the FTP server is allowed from an external network (via LAN DAU) .

return ext_net_conn: OFF | ON OFF: not allowed ON: allowed

get_ipv_six() → bool

```
# SCPI: CONFigure:DATA:CONTrol:FTP:IPVSix
value: bool = driver.configure.data.control.ftp.get_ipv_six()
```

Specifies whether the FTP server supports IPv6.

return ip_v_6_enable: OFF | ON OFF: IPv4 support only ON: support of IPv4 and IPv6

get_stype() → RsCmwDau.enums.ServiceTypeA

```
# SCPI: CONFigure:DATA:CONTrol:FTP:STYPe
value: enums.ServiceTypeA = driver.configure.data.control.ftp.get_stype()
```

Selects the FTP service type. The other CONFigure:DATA:CONTrol:FTP:... commands configure the FTP server. They are not relevant, if the service type TGENerator is selected.

return service_type: SERVer | TGENerator SERVer: an FTP server runs on the R&S CMW TGENerator: the R&S CMW acts as a traffic generator

set_auser(*anonymous: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:FTP:AUSer
driver.configure.data.control.ftp.set_auser(anonymous = False)
```

Specifies whether access to the FTP server is allowed for anonymous users.

param anonymous OFF | ON OFF: not allowed ON: allowed

set_dupload(*data_upload: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:FTP:DUPLoad
driver.configure.data.control.ftp.set_dupload(data_upload = False)
```

Specifies whether data upload to the FTP server is allowed for anonymous users.

param data_upload OFF | ON OFF: not allowed ON: allowed

set_en_connection(*ext_net_conn: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:FTP:ENConnection
driver.configure.data.control.ftp.set_en_connection(ext_net_conn = False)
```

Specifies whether access to the FTP server is allowed from an external network (via LAN DAU) .

param ext_net_conn OFF | ON OFF: not allowed ON: allowed

set_ipv_six(*ip_v_6_enable: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:FTP:IPVSix
driver.configure.data.control.ftp.set_ipv_six(ip_v_6_enable = False)
```

Specifies whether the FTP server supports IPv6.

param ip_v_6_enable OFF | ON OFF: IPv4 support only ON: support of IPv4 and IPv6

set_stype(*service_type: RsCmwDau.enums.ServiceTypeA*) → None

```
# SCPI: CONFigure:DATA:CONTrol:FTP:STYPe
driver.configure.data.control.ftp.set_stype(service_type = enums.ServiceTypeA.
↪SERVer)
```

Selects the FTP service type. The other CONFigure:DATA:CONTrol:FTP:... commands configure the FTP server. They are not relevant, if the service type TGENerator is selected.

param service_type SERVer | TGENerator SERVer: an FTP server runs on the R&S CMW TGENerator: the R&S CMW acts as a traffic generator

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.ftp.clone()
```

Subgroups

7.2.1.1.9.1 User

SCPI Commands

```
CONFigure:DATA:CONTrol:FTP:USER:ADD
CONFigure:DATA:CONTrol:FTP:USER:DELete
```

class User

User commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class AddStruct

Structure for setting input parameters. Fields:

- User: str: User name as string
- Password: str: Password as string
- Delete_Allowed: bool: OFF | ON
- Download_Allowed: bool: OFF | ON
- Upload_Allowed: bool: OFF | ON

delete(*user: str*, *password: Optional[str] = None*) → None

```
# SCPI: CONFigure:DATA:CONTrol:FTP:USER:DELete
driver.configure.data.control.ftp.user.delete(user = '1', password = '1')
```

Deletes an FTP user account.

param user FTP user name as string

param password Password of the FTP user as string - supported for backward compatibility of the command, can be omitted

set_add(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Ftp_.User.User.AddStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:FTP:USER:ADD
driver.configure.data.control.ftp.user.set_add(value = AddStruct())
```

Creates an FTP user account.

param value see the help for AddStruct structure arguments.

7.2.1.1.10 Http

SCPI Commands

```
CONFigure:DATA:CONTrol:HTTP:ENConnection
CONFigure:DATA:CONTrol:HTTP:IPVSix
```

class Http

Http commands group definition. 3 total commands, 1 Sub-groups, 2 group commands

get_en_connection() → bool

```
# SCPI: CONFigure:DATA:CONTrol:HTTP:ENConnection
value: bool = driver.configure.data.control.http.get_en_connection()
```

Specifies whether access to the internal web server is allowed from an external network (via LAN DAU) .

return ext_net_conn: OFF | ON OFF: not allowed ON: allowed

get_ipv_six() → bool

```
# SCPI: CONFigure:DATA:CONTrol:HTTP:IPVSix
value: bool = driver.configure.data.control.http.get_ipv_six()
```

Specifies whether the internal web server supports IPv6.

return ip_v_6_enable: OFF | ON OFF: IPv4 support only ON: support of IPv4 and IPv6

set_en_connection(*ext_net_conn: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:HTTP:ENConnection
driver.configure.data.control.http.set_en_connection(ext_net_conn = False)
```

Specifies whether access to the internal web server is allowed from an external network (via LAN DAU) .

param ext_net_conn OFF | ON OFF: not allowed ON: allowed

set_ipv_six(*ip_v_6_enable: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:HTTP:IPVSix
driver.configure.data.control.http.set_ipv_six(ip_v_6_enable = False)
```

Specifies whether the internal web server supports IPv6.

param ip_v_6_enable OFF | ON OFF: IPv4 support only ON: support of IPv4 and IPv6

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.http.clone()
```

Subgroups

7.2.1.1.10.1 Start

SCPI Commands

```
CONFigure:DATA:CONTrol:HTTP:STARt:INDexing
```

class Start

Start commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_indexing() → bool

```
# SCPI: CONFigure:DATA:CONTrol:HTTP:STARt:INDexing
value: bool = driver.configure.data.control.http.start.get_indexing()
```

No command help available

return index_trigger: No help available

set_indexing(*index_trigger: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:HTTP:STARt:INDexing
driver.configure.data.control.http.start.set_indexing(index_trigger = False)
```

No command help available

param index_trigger No help available

7.2.1.1.11 Epdg

class Epdg

Epdg commands group definition. 35 total commands, 10 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.clone()
```

Subgroups

7.2.1.1.11.1 Pcscf

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:PCSCf:AUTO
```

class Pcscf

Pcscf commands group definition. 4 total commands, 2 Sub-groups, 1 group commands

get_auto() → bool

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:AUTO
value: bool = driver.configure.data.control.epdg.pcscf.get_auto()
```

Enables or disables the automatic mode for the IPv4 and IPv6 type settings.

return auto: OFF | ON OFF: The configured IP types are used. ON: The IP type offered by the DUT is used.

set_auto(*auto: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:AUTO
driver.configure.data.control.epdg.pcscf.set_auto(auto = False)
```

Enables or disables the automatic mode for the IPv4 and IPv6 type settings.

param auto OFF | ON OFF: The configured IP types are used. ON: The IP type offered by the DUT is used.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.pcscf.clone()
```

Subgroups

7.2.1.1.11.2 IpvSix

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVSix:TYPE
```

class IpvSix

IpvSix commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

get_type_py() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVSix:TYPE
value: int = driver.configure.data.control.epdg.pcscf.ipvSix.get_type_py()
```

Sets the attribute type field of the P_CSCF_IP6_ADDRESS configuration attribute.

return pcscf_ip_v_6_typ: Range: 0 to 65535

set_type_py(*pcscf_ip_v_6_typ: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVSix:TYPE
driver.configure.data.control.epdg.pcscf.ipvSix.set_type_py(pcscf_ip_v_6_typ =␣
↪1)
```

Sets the attribute type field of the P_CSCF_IP6_ADDRESS configuration attribute.

param pcscf_ip_v_6_typ Range: 0 to 65535

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.pcscf.ipvSix.clone()
```

Subgroups

7.2.1.1.11.3 Address

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVSix:ADDRess:LENGth
```

class Address

Address commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_length() → RsCmwDau.enums.IpV6AddLgh

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVSix:ADDRess:LENGth
value: enums.IpV6AddLgh = driver.configure.data.control.epdg.pcscf.ipvSix.
↪address.get_length()
```

Sets the length field of the P_CSCF_IP6_ADDRESS configuration attribute.

return ip_v_6_add_lgh: L16 | L17

set_length(*ip_v_6_add_lgh: RsCmwDau.enums.IpV6AddLgh*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVSix:ADDRess:LENGth
driver.configure.data.control.epdg.pcscf.ipvSix.address.set_length(ip_v_6_add_
↪lgh = enums.IpV6AddLgh.L16)
```

Sets the length field of the P_CSCF_IP6_ADDRESS configuration attribute.

param ip_v_6_add_lgh L16 | L17

7.2.1.1.11.4 IpvFour

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVFour:TYPE
```

class IpvFour

IpvFour commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_type_py() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVFour:TYPE
value: int = driver.configure.data.control.epdg.pcscf.ipvFour.get_type_py()
```

Sets the attribute type field of the P_CSCF_IP4_ADDRESS configuration attribute.

return pcscf_ip_v_4_typ: Range: 0 to 65535

set_type_py(*pcscf_ip_v_4_typ: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:PCSCf:IPVFour:TYPE
driver.configure.data.control.epdg.pcscf.ipvFour.set_type_py(pcscf_ip_v_4_typ =␣
↪1)
```

Sets the attribute type field of the P_CSCF_IP4_ADDRESS configuration attribute.

param pcscf_ip_v_4_typ Range: 0 to 65535

7.2.1.1.11.5 Address

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:ADDRess:IPVFour
CONFigure:DATA:CONTrol:EPDG:ADDRess:IPVSix
```

class Address

Address commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_ipv_four() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ADDRess:IPVFour
value: str = driver.configure.data.control.epdg.address.get_ipv_four()
```

Specifies the IPv4 address of the ePDG.

return ip_v_4_addressess: IPv4 address string

get_ipv_six() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ADDRess:IPVSix
value: str = driver.configure.data.control.epdg.address.get_ipv_six()
```

Specifies the IPv6 address of the ePDG.

return ip_v_6_addressess: IPv6 address string

set_ipv_four(*ip_v_4_addressess: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ADDRess:IPVFour
driver.configure.data.control.epdg.address.set_ipv_four(ip_v_4_addressess = '1')
```

Specifies the IPv4 address of the ePDG.

param ip_v_4_addressess IPv4 address string

set_ipv_six(*ip_v_6_addressess: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ADDRess:IPVSix
driver.configure.data.control.epdg.address.set_ipv_six(ip_v_6_addressess = '1')
```

Specifies the IPv6 address of the ePDG.

param ip_v_6_addressess IPv6 address string

7.2.1.1.11.6 Id

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:ID:TYPE
CONFigure:DATA:CONTrol:EPDG:ID:VALue
```

class Id

Id commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_type_py() → RsCmwDau.enums.IdType

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ID:TYPE
value: enums.IdType = driver.configure.data.control.epdg.id.get_type_py()
```

Configures the type of the ePDG identification.

return id_type: IPVF | FQDN | RFC | IPVS | KEY IPVF: ID_IPv4_ADDR FQDN: ID_FQDN RFC: ID_RFC822_ADDR IPVS: ID_IPV6_ADDR KEY: ID_KEY_ID

get_value() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ID:VALue
value: str = driver.configure.data.control.epdg.id.get_value()
```

Configures the value of the ePDG identification.

return id_value: Identification as string

set_type_py(*id_type: RsCmwDau.enums.IdType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ID:TYPE
driver.configure.data.control.epdg.id.set_type_py(id_type = enums.IdType.ASND)
```

Configures the type of the ePDG identification.

param id_type IPVF | FQDN | RFC | IPVS | KEY IPVF: ID_IPv4_ADDR FQDN: ID_FQDN RFC: ID_RFC822_ADDR IPVS: ID_IPV6_ADDR KEY: ID_KEY_ID

set_value(*id_value: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ID:VALue
driver.configure.data.control.epdg.id.set_value(id_value = '1')
```

Configures the value of the ePDG identification.

param id_value Identification as string

7.2.1.1.11.7 Ike

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:IKE:ENCRyption
CONFigure:DATA:CONTrol:EPDG:IKE:PRF
CONFigure:DATA:CONTrol:EPDG:IKE:INTegrity
CONFigure:DATA:CONTrol:EPDG:IKE:DHGRoup
CONFigure:DATA:CONTrol:EPDG:IKE:LIFetime
```

class Ike

Ike commands group definition. 7 total commands, 1 Sub-groups, 5 group commands

class DhGroupStruct

Structure for reading output parameters. Fields:

- Dh_Group_1: bool: OFF | ON
- Dh_Group_2: bool: OFF | ON
- Dh_Group_5: bool: OFF | ON
- Dh_Group_14: bool: OFF | ON
- Dh_Group_15: bool: OFF | ON
- Dh_Group_16: bool: OFF | ON
- Dh_Group_17: bool: OFF | ON
- Dh_Group_18: bool: OFF | ON

class EncryptionStruct

Structure for reading output parameters. Fields:

- Aescbc: bool: OFF | ON ENCR_AES_CBC
- Encr_3_Des: bool: OFF | ON ENCR_3DES

class IntegrityStruct

Structure for reading output parameters. Fields:

- Hmac_Md_596: bool: OFF | ON AUTH_HMAC_MD5_96
- Hmac_Shai_96: bool: OFF | ON AUTH_HMAC_SHA1_96
- Aes_Xcb_96: bool: OFF | ON AUTH_AES_XCBC_96
- Sha_2256128: bool: OFF | ON AUTH_HMAC_SHA2_256_128
- Sha_2384192: bool: OFF | ON AUTH_HMAC_SHA2_384_192
- Sha_2512256: bool: OFF | ON AUTH_HMAC_SHA2_512_256

class PrfStruct

Structure for reading output parameters. Fields:

- Prfmd_5: bool: OFF | ON PRF_HMAC_MD5
- Prfsha_1: bool: OFF | ON PRF_HMAC_SHA1
- Sha_2256: bool: OFF | ON PRF_HMAC_SHA2_256
- Sha_2384: bool: OFF | ON PRF_HMAC_SHA2_384
- Sha_2512: bool: OFF | ON PRF_HMAC_SHA2_512

get_dh_group() → DhGroupStruct

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:DHGRoup
value: DhGroupStruct = driver.configure.data.control.epdg.ike.get_dh_group()
```

Selects the supported Diffie-Hellman groups for the IKEv2 protocol.

return structure: for return value, see the help for DhGroupStruct structure arguments.

get_encryption() → EncryptionStruct

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:ENCRyption
value: EncryptionStruct = driver.configure.data.control.epdg.ike.get_
↪encryption()
```

Selects the supported encryption algorithms for the IKEv2 protocol.

return structure: for return value, see the help for EncryptionStruct structure arguments.

get_integrity() → IntegrityStruct

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:INTegrity
value: IntegrityStruct = driver.configure.data.control.epdg.ike.get_integrity()
```

Selects the supported integrity protection algorithms for the IKEv2 protocol.

return structure: for return value, see the help for IntegrityStruct structure arguments.

get_lifetime() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:LIFetime
value: int = driver.configure.data.control.epdg.ike.get_lifetime()
```

No command help available

return ikesa_lifetime: No help available

get_prf() → PrfStruct

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:PRF
value: PrfStruct = driver.configure.data.control.epdg.ike.get_prf()
```

Selects the supported pseudorandom functions for the IKEv2 protocol.

return structure: for return value, see the help for PrfStruct structure arguments.

set_dh_group(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Epdg_.Ike.Ike.DhGroupStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:DHGRoup
driver.configure.data.control.epdg.ike.set_dh_group(value = DhGroupStruct())
```

Selects the supported Diffie-Hellman groups for the IKEv2 protocol.

param value see the help for DhGroupStruct structure arguments.

set_encryption(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Epdg_.Ike.Ike.EncryptionStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:ENCRyption
driver.configure.data.control.epdg.ike.set_encryption(value =␣
↪EncryptionStruct())
```

Selects the supported encryption algorithms for the IKEv2 protocol.

param value see the help for EncryptionStruct structure arguments.

set_integrity(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Epdg_.Ike.Ike.IntegrityStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:INTegrity
driver.configure.data.control.epdg.ike.set_integrity(value = IntegrityStruct())
```

Selects the supported integrity protection algorithms for the IKEv2 protocol.

param value see the help for IntegrityStruct structure arguments.

set_lifetime(*ikesa_lifetime: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:LIFetime
driver.configure.data.control.epdg.ike.set_lifetime(ikesa_lifetime = 1)
```

No command help available

param ikesa_lifetime No help available

set_prf(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Epdg_.Ike.Ike.PrfStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:PRF
driver.configure.data.control.epdg.ike.set_prf(value = PrfStruct())
```

Selects the supported pseudorandom functions for the IKEv2 protocol.

param value see the help for PrfStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.ike.clone()
```

Subgroups

7.2.1.1.11.8 Rekey

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:IKE:REKey:ENABle
CONFigure:DATA:CONTrol:EPDG:IKE:REKey:TIME
```

class Rekey

Rekey commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_enable() → bool

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:REKey:ENABle
value: bool = driver.configure.data.control.epdg.ike.rekey.get_enable()
```

No command help available

return rekey_enable: No help available

get_time() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:REKey:TIME
value: int = driver.configure.data.control.epdg.ike.rekey.get_time()
```

No command help available

return ikesa_rekeying_time: No help available

set_enable(*rekey_enable: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:REKey:ENABle
driver.configure.data.control.epdg.ike.rekey.set_enable(rekey_enable = False)
```

No command help available

param rekey_enable No help available

set_time(*ikesa_rekeying_time: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:IKE:REKey:TIME
driver.configure.data.control.epdg.ike.rekey.set_time(ikesa_rekeying_time = 1)
```

No command help available

param ikesa_rekeying_time No help available

7.2.1.1.11.9 Esp

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:ESP:ENCRyption
CONFigure:DATA:CONTrol:EPDG:ESP:INTegrity
CONFigure:DATA:CONTrol:EPDG:ESP:LIFetime
```

class Esp

Esp commands group definition. 5 total commands, 1 Sub-groups, 3 group commands

class EncryptionStruct

Structure for reading output parameters. Fields:

- Aescbc: bool: OFF | ON ENCR_AES_CBC
- Encr_3_Des: bool: OFF | ON ENCR_3DES

class IntegrityStruct

Structure for reading output parameters. Fields:

- Md_596: bool: OFF | ON AUTH_HMAC_MD5_96
- Shai_96: bool: OFF | ON AUTH_HMAC_SHA1_96
- Xcbc_96: bool: OFF | ON AUTH_AES_XCBC_96
- Sha_256: bool: OFF | ON AUTH_HMAC_SHA2_256_128
- Sha_384: bool: OFF | ON AUTH_HMAC_SHA2_384_192
- Sha_512: bool: OFF | ON AUTH_HMAC_SHA2_512_256

get_encryption() → EncryptionStruct

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:ENCRyption
value: EncryptionStruct = driver.configure.data.control.epdg.esp.get_
↪encryption()
```

Selects the supported encryption algorithms for the ESP protocol.

return structure: for return value, see the help for EncryptionStruct structure arguments.

get_integrity() → IntegrityStruct

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:INTegrity
value: IntegrityStruct = driver.configure.data.control.epdg.esp.get_integrity()
```

Selects the supported integrity protection algorithms for the ESP protocol.

return structure: for return value, see the help for IntegrityStruct structure arguments.

get_lifetime() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:LIFetime
value: int = driver.configure.data.control.epdg.esp.get_lifetime()
```

No command help available

return espsa_lifetime: No help available

set_encryption(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Epdg_.Esp.Esp.EncryptionStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:ENCRyption
driver.configure.data.control.epdg.esp.set_encryption(value =␣
↪EncryptionStruct())
```

Selects the supported encryption algorithms for the ESP protocol.

param value see the help for EncryptionStruct structure arguments.

set_integrity(*value: RsCmwDau.Implementations.Configure_.Data_.Control_.Epdg_.Esp.Esp.IntegrityStruct*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:INTegrity
driver.configure.data.control.epdg.esp.set_integrity(value = IntegrityStruct())
```

Selects the supported integrity protection algorithms for the ESP protocol.

param value see the help for IntegrityStruct structure arguments.

set_lifetime(*espsa_lifetime: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:LIFetime
driver.configure.data.control.epdg.esp.set_lifetime(espsa_lifetime = 1)
```

No command help available

param espsa_lifetime No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.esp.clone()
```

Subgroups

7.2.1.1.11.10 Rekey

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:ESP:REKey:ENABle
CONFigure:DATA:CONTrol:EPDG:ESP:REKey:TIME
```

class Rekey

Rekey commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_enable() → bool

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:REKey:ENABle
value: bool = driver.configure.data.control.epdg.esp.rekey.get_enable()
```

No command help available

return rekey_enable: No help available

get_time() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:REKey:TIME
value: int = driver.configure.data.control.epdg.esp.rekey.get_time()
```

No command help available

return espsa_rekeying_time: No help available

set_enable(*rekey_enable: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:REKey:ENABle
driver.configure.data.control.epdg.esp.rekey.set_enable(rekey_enable = False)
```

No command help available

param rekey_enable No help available

set_time(*espsa_rekeying_time: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:ESP:REKey:TIME
driver.configure.data.control.epdg.esp.rekey.set_time(espsa_rekeying_time = 1)
```

No command help available

param espsa_rekeying_time No help available

7.2.1.1.11.11 Dpd

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:DPD:ENABle
CONFigure:DATA:CONTrol:EPDG:DPD:INTerval
CONFigure:DATA:CONTrol:EPDG:DPD:TIMeout
```

class Dpd

Dpd commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_enable() → bool

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:DPD:ENABle
value: bool = driver.configure.data.control.epdg.dpd.get_enable()
```

Enables dead peer detection.

return dpd_enable: OFF | ON

get_interval() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:DPD:INTerval
value: int = driver.configure.data.control.epdg.dpd.get_interval()
```

Configures the inactivity time interval for dead peer detection.

return dpd_interval: Range: 1 s to 100 s, Unit: s

get_timeout() → int

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:DPD:TIMeout
value: int = driver.configure.data.control.epdg.dpd.get_timeout()
```

Configures the no answer timeout for dead peer detection.

return dpd_timeout: Range: 1 s to 10E+3 s, Unit: s

set_enable(*dpd_enable: bool*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:DPD:ENABle
driver.configure.data.control.epdg.dpd.set_enable(dpd_enable = False)
```

Enables dead peer detection.

param dpd_enable OFF | ON

set_interval(*dpd_interval: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:DPD:INTerval
driver.configure.data.control.epdg.dpd.set_interval(dpd_interval = 1)
```

Configures the inactivity time interval for dead peer detection.

param dpd_interval Range: 1 s to 100 s, Unit: s

set_timeout(*dpd_timeout: int*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:DPD:TIMeout
driver.configure.data.control.epdg.dpd.set_timeout(dpd_timeout = 1)
```

Configures the no answer timeout for dead peer detection.

param dpd_timeout Range: 1 s to 10E+3 s, Unit: s

7.2.1.1.11.12 Authentic

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:AUTHentic:ALGorithm
CONFigure:DATA:CONTrol:EPDG:AUTHentic:IMSI
CONFigure:DATA:CONTrol:EPDG:AUTHentic:RAND
CONFigure:DATA:CONTrol:EPDG:AUTHentic:AMF
CONFigure:DATA:CONTrol:EPDG:AUTHentic:OPC
```

class Authentic

Authentic commands group definition. 7 total commands, 1 Sub-groups, 5 group commands

get_algorithm() → RsCmwDau.enums.AuthAlgorithm

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:ALGorithm
value: enums.AuthAlgorithm = driver.configure.data.control.epdg.authentic.get_
↪algorithm()
```

Specifies the key generation algorithm set used by the SIM.

return auth_alg: XOR | MILenage

get_amf() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:AMF
value: str = driver.configure.data.control.epdg.authentic.get_amf()
```

Specifies the authentication management field (AMF) as four-digit hexadecimal number. Leading zeros can be omitted.

return auth_amf: Range: #H0 to #HFFFF

get_imsi() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:IMSI
value: str = driver.configure.data.control.epdg.authentic.get_imsi()
```

Specifies the IMSI of the SIM card.

return auth_imsi: String value, containing 15 digits

get_opc() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:OPC
value: str = driver.configure.data.control.epdg.authentic.get_opc()
```

Specifies the key OPc as 32-digit hexadecimal number. Leading zeros can be omitted.

return auth_opc: Range: #H00000000000000000000000000000000 to #HFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

get_rand() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:RAND
value: str = driver.configure.data.control.epdg.authentic.get_rand()
```

Defines the random number RAND as 32-digit hexadecimal number. Leading zeros can be omitted.

return auth_rand: Range: #H0 to #HFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

set_algorithm(*auth_alg: RsCmwDau.enums.AuthAlgorithm*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:ALGorithm
driver.configure.data.control.epdg.authentic.set_algorithm(auth_alg = enums.
↪AuthAlgorithm.MILenage)
```

Specifies the key generation algorithm set used by the SIM.

param auth_alg XOR | MILenage

set_amf(*auth_amf: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:AMF
driver.configure.data.control.epdg.authentic.set_amf(auth_amf = r1)
```

Specifies the authentication management field (AMF) as four-digit hexadecimal number. Leading zeros can be omitted.

param auth_amf Range: #H0 to #HFFFF

set_imsi(*auth_imsi: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:IMSI
driver.configure.data.control.epdg.authentic.set_imsi(auth_imsi = '1')
```

Specifies the IMSI of the SIM card.

param auth_imsi String value, containing 15 digits

set_opc(*auth_opc: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:OPC
driver.configure.data.control.epdg.authentic.set_opc(auth_opc = r1)
```

Specifies the key OPc as 32-digit hexadecimal number. Leading zeros can be omitted.

param auth_opc Range: #H00000000000000000000000000000000 to #HFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

set_rand(*auth_rand: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:RAND
driver.configure.data.control.epdg.authentic.set_rand(auth_rand = r1)
```

Defines the random number RAND as 32-digit hexadecimal number. Leading zeros can be omitted.

param auth_rand Range: #H0 to #HFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.authentic.clone()
```

Subgroups

7.2.1.1.11.13 Key

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:AUTHentic:KEY:TYPE
CONFigure:DATA:CONTrol:EPDG:AUTHentic:KEY
```

class Key

Key commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_type_py() → RsCmwDau.enums.KeyType

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:KEY:TYPE
value: enums.KeyType = driver.configure.data.control.epdg.authentic.key.get_
↪type_py()
```

Selects the key type to be used with the MILENAGE algorithm set. Currently, only OPc is supported.

return auth_key_type: OPC

get_value() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:KEY
value: str = driver.configure.data.control.epdg.authentic.key.get_value()
```

Defines the secret key K as 32-digit hexadecimal number. Leading zeros can be omitted.

return auth_key: Range: #H0 to #HFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

set_type_py(*auth_key_type: RsCmwDau.enums.KeyType*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:KEY:TYPE
driver.configure.data.control.epdg.authentic.key.set_type_py(auth_key_type =␣
↪enums.KeyType.OP)
```

Selects the key type to be used with the MILENAGE algorithm set. Currently, only OPc is supported.

param auth_key_type OPC

set_value(*auth_key: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:AUTHentic:KEY
driver.configure.data.control.epdg.authentic.key.set_value(auth_key = r1)
```

Defines the secret key K as 32-digit hexadecimal number. Leading zeros can be omitted.

param auth_key Range: #H0 to #HFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

7.2.1.1.11.14 Certificate

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:CERTificate:KEY
CONFigure:DATA:CONTrol:EPDG:CERTificate:ENABle
CONFigure:DATA:CONTrol:EPDG:CERTificate:CERTificate
```

class Certificate

Certificate commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_certificate() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CERTificate:CERTificate
value: str = driver.configure.data.control.epdg.certificate.get_certificate()
```

Selects the server certificate file to be used for SSL.

return certificate_server_file: No help available

get_enable() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CERTificate:ENABle
value: str = driver.configure.data.control.epdg.certificate.get_enable()
```

Enables the usage of certificates for SSL.

return certificate_key_enable: No help available

get_key() → str

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CERTificate:KEY
value: str = driver.configure.data.control.epdg.certificate.get_key()
```

Selects the server key file to be used for SSL.

return certificate_key_file: Filename as string, without path

set_certificate(*certificate_server_file: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CERTificate:CERTificate
driver.configure.data.control.epdg.certificate.set_certificate(certificate_
↪server_file = '1')
```

Selects the server certificate file to be used for SSL.

param certificate_server_file Filename as string, without path

set_enable(*certificate_key_enable: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CERTificate:ENABle
driver.configure.data.control.epdg.certificate.set_enable(certificate_key_
↪enable = '1')
```

Enables the usage of certificates for SSL.

param certificate_key_enable OFF | ON

set_key(*certificate_key_file: str*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CERTificate:KEY
driver.configure.data.control.epdg.certificate.set_key(certificate_key_file = '1
↪')
```

Selects the server key file to be used for SSL.

param certificate_key_file Filename as string, without path

7.2.1.1.11.15 Connections

class Connections

Connections commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.connections.clone()
```

Subgroups

7.2.1.1.11.16 Imsi<Imsi>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.control.epdg.connections.imsi.repcap_imsi_get()
driver.configure.data.control.epdg.connections.imsi.repcap_imsi_set(repcap.Imsi.Ix1)
```

class Imsi

Imsi commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Imsi, default value after init: Imsi.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.connections.imsi.clone()
```

Subgroups

7.2.1.1.11.17 Apn<AccPointName>

RepCap Settings

```
# Range: Nr1 .. Nr15
rc = driver.configure.data.control.epdg.connections.imsi.apn.repcap_accPointName_get()
driver.configure.data.control.epdg.connections.imsi.apn.repcap_accPointName_set(repcap.
↪AccPointName.Nr1)
```

class Apn

Apn commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: AccPointName, default value after init: AccPointName.Nr1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.connections.imsi.apn.clone()
```

Subgroups

7.2.1.1.11.18 Release

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:CONNections:IMSI<Imsi>:APN<AccPointName>:RELease
```

class Release

Release commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*imsi=<Imsi.Default: -1>*, *accPointName=<AccPointName.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CONNections:IMSI<Suffix>:APN<APNSuffix>
↪:RELease
value: bool = driver.configure.data.control.epdg.connections.imsi.apn.release.
↪get(imsi = repcap.Imsi.Default, accPointName = repcap.AccPointName.Default)
```

No command help available

param imsi optional repeated capability selector. Default value: Ix1 (settable in the interface 'Imsi')

param accPointName optional repeated capability selector. Default value: Nr1 (settable in the interface 'Apn')

return release: No help available

set(*release: bool*, *imsi=<Imsi.Default: -1>*, *accPointName=<AccPointName.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CONNections:IMSI<Suffix>:APN<APNSuffix>
↪:RELease
driver.configure.data.control.epdg.connections.imsi.apn.release.set(release =␣
↪False, imsi = repcap.Imsi.Default, accPointName = repcap.AccPointName.Default)
```

No command help available

param release No help available

param imsi optional repeated capability selector. Default value: Ix1 (settable in the interface 'Imsi')

param accPointName optional repeated capability selector. Default value: Nr1 (settable in the interface 'Apn')

7.2.1.1.11.19 Clean

class Clean

Clean commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.clean.clone()
```

Subgroups

7.2.1.1.11.20 General

class General

General commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.control.epdg.clean.general.clone()
```

Subgroups

7.2.1.1.11.21 Info

SCPI Commands

```
CONFigure:DATA:CONTrol:EPDG:CLEan:GENeral:INFO
```

class Info

Info commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CLEan:GENeral:INFO
driver.configure.data.control.epdg.clean.general.info.set()
```

Clears the 'ePDG Event Log' area.

set_with_opc() → None

```
# SCPI: CONFigure:DATA:CONTrol:EPDG:CLEan:GENeral:INFO
driver.configure.data.control.epdg.clean.general.info.set_with_opc()
```

Clears the 'ePDG Event Log' area.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.2 Measurement

SCPI Commands

```
CONFigure:DATA:MEASurement:IPConn
```

class Measurement

Measurement commands group definition. 117 total commands, 12 Sub-groups, 1 group commands

get_ip_connect() → bool

```
# SCPI: CONFigure:DATA:MEASurement:IPConn
value: bool = driver.configure.data.measurement.get_ip_connect()
```

No command help available

return ip_on: No help available

set_ip_connect(*ip_on: bool*) → None

```
# SCPI: CONFigure:DATA:MEASurement:IPConn
driver.configure.data.measurement.set_ip_connect(ip_on = False)
```

No command help available

param ip_on No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.clone()
```

Subgroups

7.2.1.2.1 IpAnalysis

class IpAnalysis
IpAnalysis commands group definition. 28 total commands, 8 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.clone()
```

Subgroups

7.2.1.2.1.1 IpcSecurity

class IpcSecurity
IpcSecurity commands group definition. 10 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ipcSecurity.clone()
```

Subgroups

7.2.1.2.1.2 Kyword

class Kyword
Kyword commands group definition. 3 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ipcSecurity.keyword.clone()
```

Subgroups

7.2.1.2.1.3 Search

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:KYWord:SEARch:IMPort
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:KYWord:SEARch:ADD
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:KYWord:SEARch:DELete
```

class Search

Search commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

delete(*keyword: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:KYWord:SEARch:DELete
driver.configure.data.measurement.ipAnalysis.ipcSecurity.keyword.search.
↪delete(keyword = '1')
```

Deletes a selected keyword from the keyword list.

param keyword Keyword to be deleted, as string

set_add(*keyword: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:KYWord:SEARch:ADD
driver.configure.data.measurement.ipAnalysis.ipcSecurity.keyword.search.set_
↪add(keyword = '1')
```

Adds an entry to the keyword list.

param keyword New keyword as string

set_import_py(*filename: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:KYWord:SEARch:IMPort
driver.configure.data.measurement.ipAnalysis.ipcSecurity.keyword.search.set_
↪import_py(filename = '1')
```

Imports a list of keywords from a text file.

param filename File name as string, including the extension. The file must be available in the directory ip_analysis of the samba share.

7.2.1.2.1.4 PrtScan

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:TIMeout
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:STOP
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:RANGe
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:DESTip
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:STARt
```

class PrtScan

PrtScan commands group definition. 7 total commands, 2 Sub-groups, 5 group commands

class RangeStruct

Structure for reading output parameters. Fields:

- Range_From: int: Lower end of the range Range: 0 to 65.535E+3
- Range_To: int: Upper end of the range Range: 0 to 65.535E+3

get_dest_ip() → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:DESTip
value: str = driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.
↪get_dest_ip()
```

Configures the IP address of the destination to be scanned.

return dst_ip: String containing the IP address

get_range() → RangeStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:RANGe
value: RangeStruct = driver.configure.data.measurement.ipAnalysis.ipcSecurity.
↪prtScan.get_range()
```

Defines the port range to be scanned.

return structure: for return value, see the help for RangeStruct structure arguments.

get_timeout() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:TIMeout
value: int = driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.
↪get_timeout()
```

Configures a timeout in milliseconds, for waiting for an answer from the DUT during the port scan.

return timeout: Range: 0 to 5000

set_dest_ip(*dst_ip: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:DESTip
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.set_dest_
↪ip(dst_ip = '1')
```

Configures the IP address of the destination to be scanned.

param dst_ip String containing the IP address

set_range(*value: RsCmwDau.Implementations.Configure_.Data_.Measurement_.IpAnalysis_.IpcSecurity_.PrtScan.PrtScan.RangeStruct*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:RANGe
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.set_
↪range(value = RangeStruct())
```

Defines the port range to be scanned.

param value see the help for RangeStruct structure arguments.

set_timeout(*timeout: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:TIMeout
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.set_
↪timeout(timeout = 1)
```

Configures a timeout in milliseconds, for waiting for an answer from the DUT during the port scan.

param timeout Range: 0 to 5000

start() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:STARt
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.start()
```

Initiates a port scan.

start_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:STARt
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.start_with_
↪opc()
```

Initiates a port scan.

Same as start, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:PRTScan:STOP
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.stop()
```

Aborts a port scan.

stop_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:PRTScan:STOP
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.stop_with_opc()
```

Aborts a port scan.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.clone()
```

Subgroups

7.2.1.2.1.5 Clean

class Clean

Clean commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.clean.clone()
```

Subgroups

7.2.1.2.1.6 General

class General

General commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.clean.general.
↪clone()
```

Subgroups

7.2.1.2.1.7 Info

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:PRTScan:CLEan:GENeral:INFO
```

class Info

Info commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:CLEan:GENeral:INFO
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.clean.general.
↪info.set()
```

Clears the event log.

set_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:CLEan:GENeral:INFO
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.clean.general.
↪info.set_with_opc()
```

Clears the event log.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.2.1.8 Layer

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan:LAYer:PROTocol
```

class Layer

Layer commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_protocol() → RsCmwDau.enums.Protocol

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:LAYer:PROTocol
value: enums.Protocol = driver.configure.data.measurement.ipAnalysis.
↪ipcSecurity.prtScan.layer.get_protocol()
```

Selects the protocol to be considered for the port scan.

return lyr_protocol: TCP | UDP

set_protocol(*lyr_protocol: RsCmwDau.enums.Protocol*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:PRTScan:LAYer:PROTocol
driver.configure.data.measurement.ipAnalysis.ipcSecurity.prtScan.layer.set_
↪protocol(lyr_protocol = enums.Protocol.TCP)
```

Selects the protocol to be considered for the port scan.

param lyr_protocol TCP | UDP

7.2.1.2.1.9 TcpAnalysis

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:RTTThreshold
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:TOTHreshold
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:TRTHreshold
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:TWSThreshold
```

class TcpAnalysis

TcpAnalysis commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

get_rtt_threshold() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:RTTThreshold
value: int = driver.configure.data.measurement.ipAnalysis.tcpAnalysis.get_rtt_
↪threshold()
```

Defines a threshold (upper limit) for the round-trip time (RTT) .

return threshold: Range: 0 ms to 200 ms, Unit: ms

get_to_threshold() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:TOTHreshold
value: float = driver.configure.data.measurement.ipAnalysis.tcpAnalysis.get_to_
↪threshold()
```

No command help available

return threshold: No help available

get_tr_threshold() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:TRTHreshold
value: float = driver.configure.data.measurement.ipAnalysis.tcpAnalysis.get_tr_
↪threshold()
```

Defines a threshold (upper limit) for TCP retransmissions as percentage of all transmissions.

return threshold: Range: 0 % to 100 %, Unit: %

get_tws_threshold() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:TWSThreshold
value: float = driver.configure.data.measurement.ipAnalysis.tcpAnalysis.get_tws_
↪threshold()
```

Defines a threshold (upper limit) for the current TCP window size as percentage of the negotiated maximum window size.

return threshold: Range: 0 % to 100 %, Unit: %

set_rtt_threshold(*threshold: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:RTTThreshold
driver.configure.data.measurement.ipAnalysis.tcpAnalysis.set_rtt_
↪threshold(threshold = 1)
```

Defines a threshold (upper limit) for the round-trip time (RTT) .

param threshold Range: 0 ms to 200 ms, Unit: ms

set_to_threshold(*threshold: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:TOTHreshold
driver.configure.data.measurement.ipAnalysis.tcpAnalysis.set_to_
↪threshold(threshold = 1.0)
```

No command help available

param threshold No help available

set_tr_threshold(*threshold: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:TRTHreshold
driver.configure.data.measurement.ipAnalysis.tcpAnalysis.set_tr_
↪threshold(threshold = 1.0)
```

Defines a threshold (upper limit) for TCP retransmissions as percentage of all transmissions.

param threshold Range: 0 % to 100 %, Unit: %

set_tws_threshold(*threshold: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:TWSThreshold
driver.configure.data.measurement.ipAnalysis.tcpAnalysis.set_tws_
↪threshold(threshold = 1.0)
```

Defines a threshold (upper limit) for the current TCP window size as percentage of the negotiated maximum window size.

param threshold Range: 0 % to 100 %, Unit: %

7.2.1.2.1.10 FilterPy

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:FILTer:CONNections
```

class FilterPy

FilterPy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_connections() → RsCmwDau.enums.FilterConnect

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:FILTer:CONNections
value: enums.FilterConnect = driver.configure.data.measurement.ipAnalysis.
↪filterPy.get_connections()
```

Configures a flow filter criterion for the connection state.

return filter_conn: OPEN | CLOSed | BOTH Evaluate only open connections, only closed connections or open and closed connections.

set_connections(*filter_conn: RsCmwDau.enums.FilterConnect*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:FILTer:CONNections
driver.configure.data.measurement.ipAnalysis.filterPy.set_connections(filter_
↪conn = enums.FilterConnect.BOTH)
```

Configures a flow filter criterion for the connection state.

param filter_conn OPEN | CLOSed | BOTH Evaluate only open connections, only closed connections or open and closed connections.

7.2.1.2.1.11 IpConnect

class IpConnect

IpConnect commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ipConnect.clone()
```

Subgroups

7.2.1.2.1.12 FilterPy

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPConnect:FILTer:EXTension
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:IPConnect:FILTer
```

class FilterPy

FilterPy commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ExtensionStruct

Structure for reading output parameters. Fields:

- Filter_1_On_Off: bool: OFF | ON ON: filter line 1 enabled OFF: filter line 1 disabled
- Filter_1_Type: enums.FilterType: FLOWid | IPADd | L4PR | L7PRotocol | APPL | CTRY | SRCP | DSTP Selects the property to be checked by filter line 1. FLOWid: flow IDs IPADd: IP addresses L4PR: L4 protocol L7PRotocol: L7 protocol APPL: application CTRY: country SRCP: source port DSTP: destination port
- Filter_1_String: str: Single string, containing all filter criteria for filter line 1. For rules, see 'Filter expressions'.
- Filter_2_On_Off: bool: OFF | ON ON: filter line 2 enabled OFF: filter line 2 disabled
- Filter_2_Type: enums.FilterType: FLOWid | IPADd | L4PR | L7PRotocol | APPL | CTRY | SRCP | DSTP Selects the property to be checked by filter line 2.
- Filter_2_String: str: Single string, containing all filter criteria for filter line 2.
- Filter_3_On_Off: bool: OFF | ON ON: filter line 3 enabled OFF: filter line 3 disabled
- Filter_3_Type: enums.FilterType: FLOWid | IPADd | L4PR | L7PRotocol | APPL | CTRY | SRCP | DSTP Selects the property to be checked by filter line 3.
- Filter_3_String: str: Single string, containing all filter criteria for filter line 3.
- Filter_4_On_Off: bool: OFF | ON ON: filter line 4 enabled OFF: filter line 4 disabled
- Filter_4_Type: enums.FilterType: FLOWid | IPADd | L4PR | L7PRotocol | APPL | CTRY | SRCP | DSTP Selects the property to be checked by filter line 4.
- Filter_4_String: str: Single string, containing all filter criteria for filter line 4.

class ValueStruct

Structure for reading output parameters. Fields:

- Filter_Type: enums.FilterType: No parameter help available
- Filter_String: str: No parameter help available

get_extension() → ExtensionStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPConnect:FILTer:EXTension
value: ExtensionStruct = driver.configure.data.measurement.ipAnalysis.ipConnect.
↪filterPy.get_extension()
```

Configures a flow filter for IP analysis results. For views supporting the filter, the evaluated set of flows is restricted according to the filter settings. The filter combines all enabled filter lines via AND. You can configure up to four filter lines. If you skip setting parameters, the related filter lines are not modified. A query returns all parameters, including the optional ones.

return structure: for return value, see the help for ExtensionStruct structure arguments.

get_value() → ValueStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:IPConnect:FILTer
value: ValueStruct = driver.configure.data.measurement.ipAnalysis.ipConnect.
↪filterPy.get_value()
```

No command help available

return structure: for return value, see the help for ValueStruct structure arguments.

set_extension(*value: RsCmw-Dau.Implementations.Configure_.Data_.Measurement_.IpAnalysis_.IpConnect_.FilterPy.FilterPy.ExtensionSt*
→ None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>
↪:IPANalysis:IPConnect:FILTer:EXTension
driver.configure.data.measurement.ipAnalysis.ipConnect.filterPy.set_
↪extension(value = ExtensionStruct())
```

Configures a flow filter for IP analysis results. For views supporting the filter, the evaluated set of flows is restricted according to the filter settings. The filter combines all enabled filter lines via AND. You can configure up to four filter lines. If you skip setting parameters, the related filter lines are not modified. A query returns all parameters, including the optional ones.

param value see the help for ExtensionStruct structure arguments.

set_value(*value: RsCmw-Dau.Implementations.Configure_.Data_.Measurement_.IpAnalysis_.IpConnect_.FilterPy.FilterPy.ValueStruct*)
→ None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:IPConnect:FILTer
driver.configure.data.measurement.ipAnalysis.ipConnect.filterPy.set_value(value␣
↪= ValueStruct())
```

No command help available

param value see the help for ValueStruct structure arguments.

7.2.1.2.1.13 Result

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:RESult:IPCS
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:RESult:ALL
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:RESult:TCPanalysis
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:RESult:IPConnect
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:RESult:DPCP
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:RESult:FTTRigger
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:RESult:VOIMs
```

class Result

Result commands group definition. 7 total commands, 0 Sub-groups, 7 group commands

class AllStruct

Structure for reading output parameters. Fields:

- Tcp_Analysis: bool: OFF | ON 'TCP Analysis' view OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view
- Ip_Connect: bool: OFF | ON 'IP Connectivity' view
- Dpcp: bool: OFF | ON 'Data Pie Charts' view
- Ft_Trigger: bool: OFF | ON 'Flow Throughput and Event Trigger' view
- Vo_Ims: bool: OFF | ON 'Voice over IMS' view
- Ipc_Security: bool: OFF | ON 'IP Connection Security' view

get_all() → AllStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult[:ALL]
value: AllStruct = driver.configure.data.measurement.ipAnalysis.result.get_all()
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. This command combines all other CONFigure:DATA:MEAS<i>:IPANalysis:RESult... commands.

return structure: for return value, see the help for AllStruct structure arguments.

get_dpcp() → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:DPCP
value: bool = driver.configure.data.measurement.ipAnalysis.result.get_dpcp()
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

return enable: OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

get_ft_trigger() → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:FTTRigger
value: bool = driver.configure.data.measurement.ipAnalysis.result.get_ft_
↪trigger()
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

return enable: OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

get_ip_connect() → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:IPConnect
value: bool = driver.configure.data.measurement.ipAnalysis.result.get_ip_
↪connect()
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

return enable: OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

get_ipcs() → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:IPCS
value: bool = driver.configure.data.measurement.ipAnalysis.result.get_ipcs()
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

return enable: OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

get_tcp_analysis() → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:TCPanalysis
value: bool = driver.configure.data.measurement.ipAnalysis.result.get_tcp_
↪analysis()
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

return enable: OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

get_vo_ims() → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:VOIMs
value: bool = driver.configure.data.measurement.ipAnalysis.result.get_vo_ims()
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

return enable: OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

set_all(*value: RsCmwDau.Implementations.Configure_.Data_.Measurement_.IpAnalysis_.Result.Result.AllStruct*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult[:ALL]
driver.configure.data.measurement.ipAnalysis.result.set_all(value = AllStruct())
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. This command combines all other CONFigure:DATA:MEAS<i>:IPANalysis:RESult... commands.

param value see the help for AllStruct structure arguments.

set_dpcp(*enable: bool*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:DPCP
driver.configure.data.measurement.ipAnalysis.result.set_dpcp(enable = False)
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

param enable OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

set_ft_trigger(*enable: bool*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:FTTRigger
driver.configure.data.measurement.ipAnalysis.result.set_ft_trigger(enable =␣
↪False)
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

param enable OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

set_ip_connect(*enable: bool*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:IPConnect
driver.configure.data.measurement.ipAnalysis.result.set_ip_connect(enable =␣
↪False)
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

param enable OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

set_ipcs(*enable: bool*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:IPCS
driver.configure.data.measurement.ipAnalysis.result.set_ipcs(enable = False)
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

param enable OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

set_tcp_analysis(*enable: bool*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:TCPanalysis
driver.configure.data.measurement.ipAnalysis.result.set_tcp_analysis(enable =␣
↪False)
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

param enable OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

set_vo_ims(*enable: bool*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:RESult:VOIMs
driver.configure.data.measurement.ipAnalysis.result.set_vo_ims(enable = False)
```

Enables or disables the display of the individual detailed views and the evaluation of the related results. The mnemonic after 'RESult' denotes the view: 'TCP Analysis', 'IP Connectivity', 'Data Pie Charts', 'Voice over IMS', 'IP Connection Security' and 'Flow Throughput and Event Trigger'.

param enable OFF | ON OFF: Do not evaluate results, hide the view ON: Evaluate results and show the view

7.2.1.2.1.14 FtTrigger

class FtTrigger

FtTrigger commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ftTrigger.clone()
```

Subgroups

7.2.1.2.1.15 Trace<Trace>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.configure.data.measurement.ipAnalysis.ftTrigger.trace.repcap_trace_get()
driver.configure.data.measurement.ipAnalysis.ftTrigger.trace.repcap_trace_set(repcap.
↪Trace.Ix1)
```

class Trace

Trace commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.ftTrigger.trace.clone()
```

Subgroups

7.2.1.2.1.16 TflowId

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:FTTRigger:TRACe<Trace>:TFLowid
```

class TflowId

TflowId commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*trace*=<*Trace.Default: -1*>) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:FTTRigger:TRACe
↪<TraceIndex>:TFLowid
value: int = driver.configure.data.measurement.ipAnalysis.ftTrigger.trace.
↪tflowId.get(trace = repcap.Trace.Default)
```

Assigns a connection (flow ID) to a trace index.

param trace optional repeated capability selector. Default value: Ix1 (settable in the interface 'Trace')

return flow_id: Flow ID of the connection to be assigned to the trace index To assign all connections matching the flow filter criteria, set the value 0.

set(*flow_id: int*, *trace=<Trace.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:FTTRigger:TRACe
↪<TraceIndex>:TFLowid
driver.configure.data.measurement.ipAnalysis.ftTrigger.trace.tflowId.set(flow_
↪id = 1, trace = repcap.Trace.Default)
```

Assigns a connection (flow ID) to a trace index.

param flow_id Flow ID of the connection to be assigned to the trace index To assign all connections matching the flow filter criteria, set the value 0.

param trace optional repeated capability selector. Default value: Ix1 (settable in the interface 'Trace')

7.2.1.2.1.17 ExportDb

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:EXPortdb
```

class ExportDb

ExportDb commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:EXPortdb
driver.configure.data.measurement.ipAnalysis.exportDb.set()
```

Stores the IP analysis result database to a JSON file on the DAU system drive.

set_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:EXPortdb
driver.configure.data.measurement.ipAnalysis.exportDb.set_with_opc()
```

Stores the IP analysis result database to a JSON file on the DAU system drive.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.2.1.18 Dpcp

class Dpcp

Dpcp commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipAnalysis.dpcp.clone()
```

Subgroups

7.2.1.2.1.19 DpLayer

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:DPCP:DPLayer:LAYer
```

class DpLayer

DpLayer commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_layer() → RsCmwDau.enums.Layer

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPLayer:LAYer
value: enums.Layer = driver.configure.data.measurement.ipAnalysis.dpcp.dpLayer.
↪get_layer()
```

Selects an analysis layer for the 'Data per Layer' pie chart view.

return layer: FEATure | APP | L7 | L4 | L3

set_layer(*layer: RsCmwDau.enums.Layer*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPLayer:LAYer
driver.configure.data.measurement.ipAnalysis.dpcp.dpLayer.set_layer(layer =␣
↪enums.Layer.APP)
```

Selects an analysis layer for the 'Data per Layer' pie chart view.

param layer FEATure | APP | L7 | L4 | L3

7.2.1.2.1.20 DpApplic

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPANalysis:DPCP:DPAPplic:APP
```

class DpApplic

DpApplic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_app() → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPAPplic:APP
value: str = driver.configure.data.measurement.ipAnalysis.dpcp.dpApplic.get_
↪app()
```

Selects a layer of the 'Data per Application' pie chart view. You can navigate from the current layer to the next lower or higher layer. The initial current layer is the application layer. The lower layers are layer 7, layer 4 and layer 3. To query the entries (strings) of the current layer, see method RsCmwDau.Data.Measurement.IpAnalysis.Dpcp.DpApplic.fetch.

return app_selected: String with an entry of the current layer: Navigates to the next lower layer for this entry 'Back' or string unknown at the current layer: Navigates back to the next higher layer

set_app(*app_selected: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPAPplic:APP
driver.configure.data.measurement.ipAnalysis.dpcp.dpApplic.set_app(app_selected␣
↪= '1')
```

Selects a layer of the 'Data per Application' pie chart view. You can navigate from the current layer to the next lower or higher layer. The initial current layer is the application layer. The lower layers are layer 7, layer 4 and layer 3. To query the entries (strings) of the current layer, see method RsCmwDau.Data.Measurement.IpAnalysis.Dpcp.DpApplic.fetch.

param app_selected String with an entry of the current layer: Navigates to the next lower layer for this entry 'Back' or string unknown at the current layer: Navigates back to the next higher layer

7.2.1.2.2 Throughput

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:THRoughput:MCOunt
```

class Throughput

Throughput commands group definition. 6 total commands, 1 Sub-groups, 1 group commands

get_mcount() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:MCOunt
value: int = driver.configure.data.measurement.throughput.get_mcount()
```

Specifies the total number of overall throughput results to be measured.

return max_count: Range: 5 to 3600

set_mcount(*max_count: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:MCOunt
driver.configure.data.measurement.throughput.set_mcount(max_count = 1)
```

Specifies the total number of overall throughput results to be measured.

param max_count Range: 5 to 3600

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.throughput.clone()
```

Subgroups

7.2.1.2.2.1 Ran

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:THRoughput:RAN:CATaloge
CONFigure:DATA:MEASurement<MeasInstance>:THRoughput:RAN:MCOunt
CONFigure:DATA:MEASurement<MeasInstance>:THRoughput:RAN<Slot>
```

class Ran

Ran commands group definition. 5 total commands, 1 Sub-groups, 3 group commands

get(*slot*=<*Slot.Nr1: 1*>) → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN<Index>
value: str = driver.configure.data.measurement.throughput.ran.get(slot = repcap.
↪Slot.Nr1)
```

Assigns a RAN to the RAN slot number <Index>. You can query a complete list of all supported strings via the command method RsCmwDau.Configure.Data.Measurement.Throughput.Ran.cataloge.

param slot optional repeated capability selector. Default value: Nr1

return ran: String parameter, selecting a signaling application instance

get_cataloge() → List[str]

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN:CATaloge
value: List[str] = driver.configure.data.measurement.throughput.ran.get_
↪cataloge()
```

Lists all available signaling applications. You can use the returned strings in other commands to select a RAN.

return ran: Comma-separated list of all supported values. Each value is represented as a string.

get_mcount() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN:MCOunt
value: int = driver.configure.data.measurement.throughput.ran.get_mcount()
```

Specifies the total number of RAN throughput results to be measured.

return max_count: Range: 5 to 3600

set(*ran: str*, *slot=<Slot.Nr1: 1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN<Index>
driver.configure.data.measurement.throughput.ran.set(ran = '1', slot = repcap.
↪Slot.Nr1)
```

Assigns a RAN to the RAN slot number <Index>. You can query a complete list of all supported strings via the command method RsCmwDau.Configure.Data.Measurement.Throughput.Ran.cataloge.

param ran String parameter, selecting a signaling application instance

param slot optional repeated capability selector. Default value: Nr1

set_mcount(*max_count: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN:MCOunt
driver.configure.data.measurement.throughput.ran.set_mcount(max_count = 1)
```

Specifies the total number of RAN throughput results to be measured.

param max_count Range: 5 to 3600

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.throughput.ran.clone()
```

Subgroups

7.2.1.2.2.2 Trace

class Trace

Trace commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.throughput.ran.trace.clone()
```

Subgroups

7.2.1.2.2.3 Dlink<Dlink>

RepCap Settings

```
# Range: Ix1 .. Ix4
rc = driver.configure.data.measurement.throughput.ran.trace.dlink.repcap_dlink_get()
driver.configure.data.measurement.throughput.ran.trace.dlink.repcap_dlink_set(repcap.
↪Dlink.Ix1)
```

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:THRoughput:RAN:TRACe:DLINk<Dlink>
```

class Dlink

Dlink commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: Dlink, default value after init: Dlink.Ix1

get(*dlink=<Dlink.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN:TRACe:DLINk<Index>
value: bool = driver.configure.data.measurement.throughput.ran.trace.dlink.
↪get(dlink = repcap.Dlink.Default)
```

Enables or disables the downlink trace for RAN slot number <Index>.

param dlink optional repeated capability selector. Default value: Ix1 (settable in the interface 'Dlink')

return dlink_selection: OFF | ON

set(*dlink_selection: bool*, *dlink=<Dlink.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN:TRACe:DLINk<Index>
driver.configure.data.measurement.throughput.ran.trace.dlink.set(dlink_
↪selection = False, dlink = repcap.Dlink.Default)
```

Enables or disables the downlink trace for RAN slot number <Index>.

param dlink_selection OFF | ON

param dlink optional repeated capability selector. Default value: Ix1 (settable in the interface 'Dlink')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.throughput.ran.trace.dlink.clone()
```

7.2.1.2.2.4 Ulink<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.configure.data.measurement.throughput.ran.trace.ulink.repcap_slot_get()
driver.configure.data.measurement.throughput.ran.trace.ulink.repcap_slot_set(repcap.Slot.
↪Nr1)
```

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:THRoughput:RAN:TRACe:ULINk<Slot>
```

class Ulink

Ulink commands group definition. 1 total commands, 0 Sub-groups, 1 group commands Repeated Capability: Slot, default value after init: Slot.Nr1

get(*slot=<Slot.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN:TRACe:ULINk<Index>
value: bool = driver.configure.data.measurement.throughput.ran.trace.ulink.
↪get(slot = repcap.Slot.Default)
```

Enables or disables the uplink trace for RAN slot number <Index>.

param slot optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ulink')

return uplink_selection: OFF | ON

set(*uplink_selection: bool*, *slot=<Slot.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:THRoughput:RAN:TRACe:ULINk<Index>
driver.configure.data.measurement.throughput.ran.trace.ulink.set(uplink_
↪selection = False, slot = repcap.Slot.Default)
```

Enables or disables the uplink trace for RAN slot number <Index>.

param uplink_selection OFF | ON

param slot optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ulink')

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.throughput.ran.trace.ulink.clone()
```

7.2.1.2.3 Select

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:SELect:APP
CONFigure:DATA:MEASurement<MeasInstance>:SELect:THRoughput
```

class Select

Select commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_app() → RsCmwDau.enums.ApplicationType

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:SELect:APP
value: enums.ApplicationType = driver.configure.data.measurement.select.get_
↪app()
```

Selects the measurement tab to be displayed.

return application_type: OVERview | PING | IPERf | THRoughput | DNSReq | IPLogging | IPANalysis | IPReplay | AUDiodelay

get_throughput() → RsCmwDau.enums.ThroughputType

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:SELect:THRoughput
value: enums.ThroughputType = driver.configure.data.measurement.select.get_
↪throughput()
```

Selects the overall throughput tab or the RAN throughput tab for display at the GUI. This command is useful for taking screenshots via remote commands.

return throughput_type: OVERall | RAN

set_app(*application_type: RsCmwDau.enums.ApplicationType*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:SELect:APP
driver.configure.data.measurement.select.set_app(application_type = enums.
↪ApplicationType.AUDiodelay)
```

Selects the measurement tab to be displayed.

param application_type OVERview | PING | IPERf | THRoughput | DNSReq | IPLogging | IPANalysis | IPReplay | AUDiodelay

set_throughput(*throughput_type: RsCmwDau.enums.ThroughputType*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:SELect:THRoughput
driver.configure.data.measurement.select.set_throughput(throughput_type = enums.
↪ThroughputType.OVERall)
```

Selects the overall throughput tab or the RAN throughput tab for display at the GUI. This command is useful for taking screenshots via remote commands.

param throughput_type OVERall | RAN

7.2.1.2.4 Ran

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:RAN:CATaloge
CONFigure:DATA:MEASurement<MeasInstance>:RAN
```

class Ran

Ran commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

get_cataloge() → List[str]

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:RAN:CATaloge
value: List[str] = driver.configure.data.measurement.ran.get_cataloge()
```

Lists all available signaling applications. You can use the returned strings in other commands to select a RAN.

return ran: Comma-separated list of all supported values. Each value is represented as a string.

get_value() → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:RAN
value: str = driver.configure.data.measurement.ran.get_value()
```

Selects an installed signaling application instance. You can query a complete list of all supported strings via the command method RsCmwDau.Configure.Data.Measurement.Ran.cataloge.

return ran: String parameter, selecting a signaling application instance

set_value(*ran: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:RAN
driver.configure.data.measurement.ran.set_value(ran = '1')
```

Selects an installed signaling application instance. You can query a complete list of all supported strings via the command method RsCmwDau.Configure.Data.Measurement.Ran.cataloge.

param ran String parameter, selecting a signaling application instance

7.2.1.2.5 Adelay

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:ADELay:SAMPles
CONFigure:DATA:MEASurement<MeasInstance>:ADELay:SPINterval
CONFigure:DATA:MEASurement<MeasInstance>:ADELay:MSAMples
```

class Adelay

Adelay commands group definition. 3 total commands, 0 Sub-groups, 3 group commands

get_msamples() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:ADELay:MSAMples
value: int = driver.configure.data.measurement.adelay.get_msamples()
```

Configures the maximum number of samples that can be displayed in the result diagrams. The traces cover the sample range -<MaxSamples> + 1 to 0.

return max_samples: Range: 1500 to 6000

get_samples() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:ADELay:SAMPles
value: int = driver.configure.data.measurement.adelay.get_samples()
```

Queries the fixed duration of a measurement interval.

return interval: Range: 1 s , Unit: s

get_sp_interval() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:ADELay:SPINterval
value: int = driver.configure.data.measurement.adelay.get_sp_interval()
```

Queries the fixed number of measurement samples per interval.

return smpl_per_interval: Range: 50

set_msamples(*max_samples: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:ADELay:MSAMples
driver.configure.data.measurement.adelay.set_msamples(max_samples = 1)
```

Configures the maximum number of samples that can be displayed in the result diagrams. The traces cover the sample range -<MaxSamples> + 1 to 0.

param max_samples Range: 1500 to 6000

7.2.1.2.6 Ping

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:PING:TIMeout
CONFigure:DATA:MEASurement<MeasInstance>:PING:DIPaddress
CONFigure:DATA:MEASurement<MeasInstance>:PING:PSIZe
CONFigure:DATA:MEASurement<MeasInstance>:PING:PCOunt
CONFigure:DATA:MEASurement<MeasInstance>:PING:INTerval
```

class Ping

Ping commands group definition. 5 total commands, 0 Sub-groups, 5 group commands

get_dip_address() → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:DIPaddress
value: str = driver.configure.data.measurement.ping.get_dip_address()
```

Specifies the destination IP address for the ping command.

return ip_address: IPv4 or IPv6 address as string

get_interval() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:INTerval
value: float = driver.configure.data.measurement.ping.get_interval()
```

Specifies the interval between two ping requests.

return interval: Range: 0.2 s to 10 s, Unit: s

get_pcount() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:PCOunt
value: int = driver.configure.data.measurement.ping.get_pcount()
```

Specifies the number of echo request packets to be sent.

return ping_count: Range: 1 to 1000

get_psize() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:PSIZe
value: int = driver.configure.data.measurement.ping.get_psize()
```

Specifies the payload size of echo request packets.

return packet_size: Range: 0 bytes to 65507 bytes , Unit: bytes

get_timeout() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:TIMeout
value: float = driver.configure.data.measurement.ping.get_timeout()
```

Specifies a timeout for ping requests.

return timeout: Range: 1 s to 9 s, Unit: s

set_dip_address(*ip_address: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:DIPaddress
driver.configure.data.measurement.ping.set_dip_address(ip_address = '1')
```

Specifies the destination IP address for the ping command.

param ip_address IPv4 or IPv6 address as string

set_interval(*interval: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:INTerval
driver.configure.data.measurement.ping.set_interval(interval = 1.0)
```

Specifies the interval between two ping requests.

param interval Range: 0.2 s to 10 s, Unit: s

set_pcount(*ping_count: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:PCOunt
driver.configure.data.measurement.ping.set_pcount(ping_count = 1)
```

Specifies the number of echo request packets to be sent.

param ping_count Range: 1 to 1000

set_psize(*packet_size: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:PSIZe
driver.configure.data.measurement.ping.set_psize(packet_size = 1)
```

Specifies the payload size of echo request packets.

param packet_size Range: 0 bytes to 65507 bytes , Unit: bytes

set_timeout(*timeout: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:PING:TIMeout
driver.configure.data.measurement.ping.set_timeout(timeout = 1.0)
```

Specifies a timeout for ping requests.

param timeout Range: 1 s to 9 s, Unit: s

7.2.1.2.7 DnsRequests

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:DNSRequests:MICount
```

class DnsRequests

DnsRequests commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_mi_count() → int

```
# SCPI: CONFigure:DATA:MEASurement<inst>:DNSRequests:MICount
value: int = driver.configure.data.measurement.dnsRequests.get_mi_count()
```

Specifies the maximum length of the result list for DNS requests measurements. The result list is stored in a ring buffer. When it is full, the first result line is deleted whenever a new result line is added to the end.

return max_index_count: Maximum number of DNS requests in the result list Range: 1 to 1000

set_mi_count(*max_index_count: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<inst>:DNSRequests:MICount
driver.configure.data.measurement.dnsRequests.set_mi_count(max_index_count = 1)
```

Specifies the maximum length of the result list for DNS requests measurements. The result list is stored in a ring buffer. When it is full, the first result line is deleted whenever a new result line is added to the end.

param max_index_count Maximum number of DNS requests in the result list Range: 1 to 1000

7.2.1.2.8 Iperf

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:TYPE
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:TDURation
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:PSIZe
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:STYPe
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:WSIZe
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:PORT
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:LPORt
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:PROTocol
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:IPADdress
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:BITRate
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:PCONnection
```

class Iperf

Iperf commands group definition. 31 total commands, 3 Sub-groups, 11 group commands

get_bitrate() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:BITRate
value: int = driver.configure.data.measurement.iperf.get_bitrate()
```

No command help available

return bitrate: No help available

get_ip_address() → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:IPADdress
value: str = driver.configure.data.measurement.iperf.get_ip_address()
```

No command help available

return ip_address: No help available

get_lport() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:LPORt
value: int = driver.configure.data.measurement.iperf.get_lport()
```

No command help available

return listen_port: No help available

get_pconnection() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PCONnection
value: int = driver.configure.data.measurement.iperf.get_pconnection()
```

No command help available

return par_conn: No help available

get_port() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PORT
value: int = driver.configure.data.measurement.iperf.get_port()
```

No command help available

return port: No help available

get_protocol() → RsCmwDau.enums.Protocol

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PROTocol
value: enums.Protocol = driver.configure.data.measurement.iperf.get_protocol()
```

No command help available

return protocol: No help available

get_psize() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PSIZe
value: int = driver.configure.data.measurement.iperf.get_psize()
```

Defines the packet size for iperf tests.

return packet_size: Range: 40 bytes to 65507 bytes, Unit: bytes

get_stype() → RsCmwDau.enums.ServiceTypeB

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:STYPe
value: enums.ServiceTypeB = driver.configure.data.measurement.iperf.get_stype()
```

No command help available

return service_type: No help available

get_tduration() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:TDURation
value: int = driver.configure.data.measurement.iperf.get_tduration()
```

Defines the duration of the test.

return test_duration: Range: 1 s to 1E+6 s, Unit: s

get_type_py() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:TYPE
value: float = driver.configure.data.measurement.iperf.get_type_py()
```

Selects the type of iperf to be used.

return iperf_type: IPERf | IP3 | IPNat Iperf or iperf3 or iperf(NAT)

get_wsize() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:WSIZe
value: float = driver.configure.data.measurement.iperf.get_wsize()
```

No command help available

return window_size: No help available

set_bitrate(*bitrate: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:BITRate
driver.configure.data.measurement.iperf.set_bitrate(bitrate = 1)
```

No command help available

param bitrate No help available

set_ip_address(*ip_address: str*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:IPADdress
driver.configure.data.measurement.iperf.set_ip_address(ip_address = '1')
```

No command help available

param ip_address No help available

set_lport(*listen_port: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:LPORt
driver.configure.data.measurement.iperf.set_lport(listen_port = 1)
```

No command help available

param listen_port No help available

set_pconnection(*par_conn: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PCONnection
driver.configure.data.measurement.iperf.set_pconnection(par_conn = 1)
```

No command help available

param par_conn No help available

set_port(*port: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PORT
driver.configure.data.measurement.iperf.set_port(port = 1)
```

No command help available

param port No help available

set_protocol(*protocol: RsCmwDau.enums.Protocol*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PROTocol
driver.configure.data.measurement.iperf.set_protocol(protocol = enums.Protocol.
↪TCP)
```

No command help available

param protocol No help available

set_psize(*packet_size: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:PSIZe
driver.configure.data.measurement.iperf.set_psize(packet_size = 1)
```

Defines the packet size for iperf tests.

param packet_size Range: 40 bytes to 65507 bytes, Unit: bytes

set_stype(*service_type: RsCmwDau.enums.ServiceTypeB*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:STYPe
driver.configure.data.measurement.iperf.set_stype(service_type = enums.
↪ServiceTypeB.BIDirectional)
```

No command help available

param service_type No help available

set_tduration(*test_duration: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:TDURation
driver.configure.data.measurement.iperf.set_tduration(test_duration = 1)
```

Defines the duration of the test.

param test_duration Range: 1 s to 1E+6 s, Unit: s

set_type_py(*iperf_type: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:TYPE
driver.configure.data.measurement.iperf.set_type_py(iperf_type = 1.0)
```

Selects the type of iperf to be used.

param iperf_type IPERf | IP3 | IPNat Iperf or iperf3 or iperf(NAT)

set_wsize(*window_size: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:WSIZe
driver.configure.data.measurement.iperf.set_wsize(window_size = 1.0)
```

No command help available

param window_size No help available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.iperf.clone()
```

Subgroups

7.2.1.2.8.1 Server<Server>

RepCap Settings

```
# Range: Ix1 .. Ix8
rc = driver.configure.data.measurement.iperf.server.repcap_server_get()
driver.configure.data.measurement.iperf.server.repcap_server_set(repcap.Server.Ix1)
```

class Server

Server commands group definition. 5 total commands, 5 Sub-groups, 0 group commands Repeated Capability: Server, default value after init: Server.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.iperf.server.clone()
```

Subgroups

7.2.1.2.8.2 SbSize

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:SERVer<Server>:SBSize
```

class SbSize

SbSize commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*server*=<*Server.Default: -1*>) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:SBSize
value: float = driver.configure.data.measurement.iperf.server.sbSize.get(server␣
↪= repcap.Server.Default)
```

Specifies the size of the socket buffer for an iperf server instance.

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

return sb_size: Range: 0 kByte to 10240 kByte, Unit: kByte

set(*sb_size: float*, *server*=<*Server.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:SBSize
driver.configure.data.measurement.iperf.server.sbSize.set(sb_size = 1.0, server␣
↪= repcap.Server.Default)
```

Specifies the size of the socket buffer for an iperf server instance.

param sb_size Range: 0 kByte to 10240 kByte, Unit: kByte

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

7.2.1.2.8.3 Enable

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:SERVer<Server>:ENABle
```

class Enable

Enable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*server*=<*Server.Default: -1*>) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:ENABle
value: bool = driver.configure.data.measurement.iperf.server.enable.get(server␣
↪= repcap.Server.Default)
```

Activates or deactivates an iperf server instance.

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

return enable: OFF | ON

set(*enable: bool*, *server*=<*Server.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:ENABle
driver.configure.data.measurement.iperf.server.enable.set(enable = False,␣
↪server = repcap.Server.Default)
```

Activates or deactivates an iperf server instance.

param enable OFF | ON

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

7.2.1.2.8.4 Protocol

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:SERVer<Server>:PROTocol
```

class Protocol

Protocol commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*server*=<*Server.Default: -1*>) → RsCmwDau.enums.Protocol

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:PROTocol
value: enums.Protocol = driver.configure.data.measurement.iperf.server.protocol.
↪get(server = repcap.Server.Default)
```

Selects the protocol type to be used for an iperf server instance.

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

return protocol: UDP | TCP UDP: use the user datagram protocol TCP: use the transport control protocol

set(*protocol: RsCmwDau.enums.Protocol*, *server*=<*Server.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:PROTocol
driver.configure.data.measurement.iperf.server.protocol.set(protocol = enums.
↪Protocol.TCP, server = repcap.Server.Default)
```

Selects the protocol type to be used for an iperf server instance.

param protocol UDP | TCP UDP: use the user datagram protocol TCP: use the transport control protocol

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

7.2.1.2.8.5 Wsize

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:SERVer<Server>:WSIZe
```

class Wsize

Wsize commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*server*=<*Server.Default: -1*>) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:WSIZe
value: float = driver.configure.data.measurement.iperf.server.wsize.get(server␣
↪= repcap.Server.Default)
```

No command help available

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

return window_size: No help available

set(*window_size: float*, *server*=<*Server.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:WSIZe
driver.configure.data.measurement.iperf.server.wsize.set(window_size = 1.0,␣
↪server = repcap.Server.Default)
```

No command help available

param window_size No help available

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

7.2.1.2.8.6 Port

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:SERVer<Server>:PORT
```

class Port

Port commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*server*=<*Server.Default: -1*>) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:PORT
value: int = driver.configure.data.measurement.iperf.server.port.get(server =␣
↪repcap.Server.Default)
```

Defines the LAN DAU port number for an iperf server instance.

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

return port: Range: 0 to 65535

set(*port: int*, *server*=<*Server.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:SERVer<Index>:PORT
driver.configure.data.measurement.iperf.server.port.set(port = 1, server =␣
↪repcap.Server.Default)
```

Defines the LAN DAU port number for an iperf server instance.

param port Range: 0 to 65535

param server optional repeated capability selector. Default value: Ix1 (settable in the interface 'Server')

7.2.1.2.8.7 Client<Client>

RepCap Settings

```
# Range: Ix1 .. Ix8
rc = driver.configure.data.measurement.iperf.client.repcap_client_get()
driver.configure.data.measurement.iperf.client.repcap_client_set(repcap.Client.Ix1)
```

class Client

Client commands group definition. 9 total commands, 9 Sub-groups, 0 group commands Repeated Capability: Client, default value after init: Client.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.iperf.client.clone()
```

Subgroups

7.2.1.2.8.8 SbSize

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:SBSize
```

class SbSize

SbSize commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:SBSize
value: float = driver.configure.data.measurement.iperf.client.sbSize.get(client␣
↪= repcap.Client.Default)
```

Specifies the size of the socket buffer for an iperf/iperf3 client instance.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return sb_size: Range: 0 kByte to 10240 kByte, Unit: kByte

set(*sb_size: float*, *client*=<*Client.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:SBSize
driver.configure.data.measurement.iperf.client.sbSize.set(sb_size = 1.0, client␣
↪= repcap.Client.Default)
```

Specifies the size of the socket buffer for an iperf/iperf3 client instance.

param sb_size Range: 0 kByte to 10240 kByte, Unit: kByte

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.9 Enable

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:ENABle
```

class Enable

Enable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:ENABle
value: bool = driver.configure.data.measurement.iperf.client.enable.get(client␣
↪= repcap.Client.Default)
```

Activates or deactivates an iperf/iperf3 client instance.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return enable: OFF | ON

set(*enable: bool*, *client*=<*Client.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:ENABle
driver.configure.data.measurement.iperf.client.enable.set(enable = False,␣
↪client = repcap.Client.Default)
```

Activates or deactivates an iperf/iperf3 client instance.

param enable OFF | ON

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.10 Protocol

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:PROTocol
```

class Protocol

Protocol commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → RsCmwDau.enums.Protocol

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:PROTocol
value: enums.Protocol = driver.configure.data.measurement.iperf.client.protocol.
↪get(client = repcap.Client.Default)
```

Selects the protocol type to be used for an iperf/iperf3 client instance.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return protocol: UDP | TCP UDP: use the user datagram protocol TCP: use the transport control protocol

set(*protocol: RsCmwDau.enums.Protocol*, *client=<Client.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:PROTocol
driver.configure.data.measurement.iperf.client.protocol.set(protocol = enums.
↪Protocol.TCP, client = repcap.Client.Default)
```

Selects the protocol type to be used for an iperf/iperf3 client instance.

param protocol UDP | TCP UDP: use the user datagram protocol TCP: use the transport control protocol

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.11 Wsize

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:WSIZe
```

class Wsize

Wsize commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client=<Client.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:WSIZe
value: float = driver.configure.data.measurement.iperf.client.wsize.get(client␣
↪= repcap.Client.Default)
```

No command help available

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return window_size: No help available

set(*window_size: float*, *client=<Client.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:WSIZe
driver.configure.data.measurement.iperf.client.wsize.set(window_size = 1.0,␣
↪client = repcap.Client.Default)
```

No command help available

param window_size No help available

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.12 Port

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:PORT
```

class Port

Port commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:PORT
value: int = driver.configure.data.measurement.iperf.client.port.get(client =␣
↪repcap.Client.Default)
```

Defines the LAN DAU port number for an iperf/iperf3 client instance.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return port: Range: 0 to 65535

set(*port: int*, *client*=<*Client.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:PORT
driver.configure.data.measurement.iperf.client.port.set(port = 1, client =␣
↪repcap.Client.Default)
```

Defines the LAN DAU port number for an iperf/iperf3 client instance.

param port Range: 0 to 65535

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.13 IpAddress

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:IPADdress
```

class IpAddress

IpAddress commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:IPADdress
value: str = driver.configure.data.measurement.iperf.client.ipAddress.
↪get(client = repcap.Client.Default)
```

Specifies the IP address of the DUT for an iperf/iperf3 client instance.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return ip_address: String containing the IP address

set(*ip_address: str*, *client*=<*Client.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:IPADdress
driver.configure.data.measurement.iperf.client.ipAddress.set(ip_address = '1',␣
↪client = repcap.Client.Default)
```

Specifies the IP address of the DUT for an iperf/iperf3 client instance.

param ip_address String containing the IP address

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.14 Pconnection

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:PCONnection
```

class Pconnection

Pconnection commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:PCONnection
value: int = driver.configure.data.measurement.iperf.client.pconnection.
↪get(client = repcap.Client.Default)
```

Specifies the number of parallel connections for an iperf/iperf3 client instance. Only applicable for protocol type TCP.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return par_conn: Range: 1 to 4

set(*par_conn: int*, *client*=<*Client.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:PCONnection
driver.configure.data.measurement.iperf.client.pconnection.set(par_conn = 1,␣
↪client = repcap.Client.Default)
```

Specifies the number of parallel connections for an iperf/iperf3 client instance. Only applicable for protocol type TCP.

param par_conn Range: 1 to 4

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.15 Bitrate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:BITRate
```

class Bitrate

Bitrate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:BITRate
value: float = driver.configure.data.measurement.iperf.client.bitrate.
↪get(client = repcap.Client.Default)
```

Defines the maximum bit rate for an iperf/iperf3 client instance.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return bitrate: Maximum bit rate to be transferred Range: 0 bit/s to 4E+9 bit/s, Unit: bit/s

set(*bitrate: float*, *client*=<*Client.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:BITRate
driver.configure.data.measurement.iperf.client.bitrate.set(bitrate = 1.0,␣
↪client = repcap.Client.Default)
```

Defines the maximum bit rate for an iperf/iperf3 client instance.

param bitrate Maximum bit rate to be transferred Range: 0 bit/s to 4E+9 bit/s, Unit: bit/s

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.16 Reverse

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:CLIent<Client>:REVerse
```

class Reverse

Reverse commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*client*=<*Client.Default: -1*>) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:REVerse
value: bool = driver.configure.data.measurement.iperf.client.reverse.get(client␣
↪= repcap.Client.Default)
```

Enables the reverse mode for an iperf3 client instance.

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

return mode: OFF | ON ON: reverse mode OFF: normal mode

set(*mode: bool*, *client*=<*Client.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:CLIent<Index>:REVerse
driver.configure.data.measurement.iperf.client.reverse.set(mode = False, client␣
↪= repcap.Client.Default)
```

Enables the reverse mode for an iperf3 client instance.

param mode OFF | ON ON: reverse mode OFF: normal mode

param client optional repeated capability selector. Default value: Ix1 (settable in the interface 'Client')

7.2.1.2.8.17 Nat<Nat>

RepCap Settings

```
# Range: Ix1 .. Ix8
rc = driver.configure.data.measurement.iperf.nat.repcap_nat_get()
driver.configure.data.measurement.iperf.nat.repcap_nat_set(repcap.Nat.Ix1)
```

class Nat

Nat commands group definition. 6 total commands, 6 Sub-groups, 0 group commands Repeated Capability: Nat, default value after init: Nat.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.iperf.nat.clone()
```

Subgroups

7.2.1.2.8.18 SbSize

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:NAT<Nat>:SBSize
```

class SbSize

SbSize commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*nat*=<*Nat.Default: -1*>) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:SBSize
value: float = driver.configure.data.measurement.iperf.nat.sbSize.get(nat =␣
↪repcap.Nat.Default)
```

(continues on next page)

(continued from previous page)

Specifies the size of the socket buffer for an iperf(NAT) client instance.

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

return sb_size: Range: 0 kByte to 10240 kByte, Unit: kByte

set(*sb_size: float*, *nat=<Nat.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:SBSize
driver.configure.data.measurement.iperf.nat.sbSize.set(sb_size = 1.0, nat =␣
↪repcap.Nat.Default)
```

Specifies the size of the socket buffer for an iperf(NAT) client instance.

param sb_size Range: 0 kByte to 10240 kByte, Unit: kByte

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

7.2.1.2.8.19 Enable

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:NAT<Nat>:ENABle
```

class Enable

Enable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*nat=<Nat.Default: -1>*) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:ENABle
value: bool = driver.configure.data.measurement.iperf.nat.enable.get(nat =␣
↪repcap.Nat.Default)
```

Activates or deactivates an iperf(NAT) client instance.

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

return enable: OFF | ON

set(*enable: bool*, *nat=<Nat.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:ENABle
driver.configure.data.measurement.iperf.nat.enable.set(enable = False, nat =␣
↪repcap.Nat.Default)
```

Activates or deactivates an iperf(NAT) client instance.

param enable OFF | ON

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

7.2.1.2.8.20 Protocol

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:NAT<Nat>:PROTocol
```

class Protocol

Protocol commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*nat*=<*Nat.Default: -1*>) → RsCmwDau.enums.Protocol

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:PROTocol
value: enums.Protocol = driver.configure.data.measurement.iperf.nat.protocol.
↪get(nat = repcap.Nat.Default)
```

Selects the protocol type to be used for an iperf(NAT) client instance.

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

return protocol: UDP | TCP UDP: use the user datagram protocol TCP: use the transport control protocol

set(*protocol: RsCmwDau.enums.Protocol*, *nat*=<*Nat.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:PROTocol
driver.configure.data.measurement.iperf.nat.protocol.set(protocol = enums.
↪Protocol.TCP, nat = repcap.Nat.Default)
```

Selects the protocol type to be used for an iperf(NAT) client instance.

param protocol UDP | TCP UDP: use the user datagram protocol TCP: use the transport control protocol

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

7.2.1.2.8.21 Port

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:NAT<Nat>:PORT
```

class Port

Port commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*nat*=<*Nat.Default: -1*>) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:PORT
value: int = driver.configure.data.measurement.iperf.nat.port.get(nat = repcap.
↪Nat.Default)
```

Defines the LAN DAU port number for an iperf(NAT) client instance.

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

return port: Range: 0 to 65535

set(*port: int*, *nat=<Nat.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:PORT
driver.configure.data.measurement.iperf.nat.port.set(port = 1, nat = repcap.Nat.
↪Default)
```

Defines the LAN DAU port number for an iperf(NAT) client instance.

param port Range: 0 to 65535

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

7.2.1.2.8.22 Pconnection

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:NAT<Nat>:PCONnection
```

class Pconnection

Pconnection commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*nat=<Nat.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:PCONnection
value: int = driver.configure.data.measurement.iperf.nat.pconnection.get(nat =␣
↪repcap.Nat.Default)
```

Specifies the number of parallel connections for an iperf(NAT) client instance. Only applicable for protocol type TCP.

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

return par_conn: Range: 1 to 4

set(*par_conn: int*, *nat=<Nat.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:PCONnection
driver.configure.data.measurement.iperf.nat.pconnection.set(par_conn = 1, nat =␣
↪repcap.Nat.Default)
```

Specifies the number of parallel connections for an iperf(NAT) client instance. Only applicable for protocol type TCP.

param par_conn Range: 1 to 4

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

7.2.1.2.8.23 Bitrate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPERf:NAT<Nat>:BITRate
```

class Bitrate

Bitrate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*nat*=<*Nat.Default: -1*>) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:BITRate
value: float = driver.configure.data.measurement.iperf.nat.bitrate.get(nat =␣
↪repcap.Nat.Default)
```

Defines the maximum bit rate for an iperf(NAT) instance.

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

return bitrate: Maximum bit rate to be transferred Range: 0 bit/s to 4E+9 bit/s, Unit: bit/s

set(*bitrate: float*, *nat*=<*Nat.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPERf:NAT<Index>:BITRate
driver.configure.data.measurement.iperf.nat.bitrate.set(bitrate = 1.0, nat =␣
↪repcap.Nat.Default)
```

Defines the maximum bit rate for an iperf(NAT) instance.

param bitrate Maximum bit rate to be transferred Range: 0 bit/s to 4E+9 bit/s, Unit: bit/s

param nat optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nat')

7.2.1.2.9 IpLogging

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPLogging:TYPE
CONFigure:DATA:MEASurement<MeasInstance>:IPLogging:FSIZe
CONFigure:DATA:MEASurement<MeasInstance>:IPLogging:PCOunter
CONFigure:DATA:MEASurement<MeasInstance>:IPLogging:PSLength
```

class IpLogging

IpLogging commands group definition. 4 total commands, 0 Sub-groups, 4 group commands

get_fsize() → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:FSIZe
value: float = driver.configure.data.measurement.ipLogging.get_fsize()
```

Configures the maximum log file size. When this file size is reached, logging stops. The default value 0 bytes means that no limit is defined.

return file_size: Range: 0 bytes to 1E+9 bytes, Unit: bytes

get_pcounter() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:PCOunter
value: int = driver.configure.data.measurement.ipLogging.get_pcounter()
```

Configures the maximum number of IP packets to be logged. When this number of packets is reached, logging stops. The default value 0 means that no limit is defined.

return packet_counter: Range: 0 to 1E+6

get_ps_length() → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:PSLength
value: int = driver.configure.data.measurement.ipLogging.get_ps_length()
```

Configures the maximum number of bytes to be logged for each IP packet. If the packet is longer, only the specified number of bytes is logged. The remaining bytes of the packet are ignored. The default value 0 means that no limit is defined.

return pkt_snap_length: Range: 0 bytes to 65565 bytes, Unit: bytes

get_type_py() → RsCmwDau.enums.LoggingType

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:TYPE
value: enums.LoggingType = driver.configure.data.measurement.ipLogging.get_type_
↪py()
```

Selects the interface to be monitored.

return logging_type: UPIP | UPPP | LANDau | UPMulti | UIPClient UPIP: IP unicast traffic from/to the DUT UPPP: PPP encapsulated IP traffic from/to the DUT LAN-

Dau: IP traffic at the LAN DAU connector UPMulti: IP multicast traffic to the DUT UIPClient: IP traffic from/to the DUT with the DAU as client

set_fsize(*file_size: float*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:FSIZe
driver.configure.data.measurement.ipLogging.set_fsize(file_size = 1.0)
```

Configures the maximum log file size. When this file size is reached, logging stops. The default value 0 bytes means that no limit is defined.

param file_size Range: 0 bytes to 1E+9 bytes, Unit: bytes

set_pcounter(*packet_counter: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:PCOunter
driver.configure.data.measurement.ipLogging.set_pcounter(packet_counter = 1)
```

Configures the maximum number of IP packets to be logged. When this number of packets is reached, logging stops. The default value 0 means that no limit is defined.

param packet_counter Range: 0 to 1E+6

set_ps_length(*pkt_snap_length: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:PSLength
driver.configure.data.measurement.ipLogging.set_ps_length(pkt_snap_length = 1)
```

Configures the maximum number of bytes to be logged for each IP packet. If the packet is longer, only the specified number of bytes is logged. The remaining bytes of the packet are ignored. The default value 0 means that no limit is defined.

param pkt_snap_length Range: 0 bytes to 65565 bytes, Unit: bytes

set_type_py(*logging_type: RsCmwDau.enums.LoggingType*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPLogging:TYPE
driver.configure.data.measurement.ipLogging.set_type_py(logging_type = enums.
↪LoggingType.LANDau)
```

Selects the interface to be monitored.

param logging_type UPIP | UPPP | LANDau | UPMulti | UIPClient UPIP: IP unicast traffic from/to the DUT UPPP: PPP encapsulated IP traffic from/to the DUT LANDau: IP traffic at the LAN DAU connector UPMulti: IP multicast traffic to the DUT UIPClient: IP traffic from/to the DUT with the DAU as client

7.2.1.2.10 IpReplay

class IpReplay
IpReplay commands group definition. 6 total commands, 6 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.ipReplay.clone()
```

Subgroups

7.2.1.2.10.1 CreateList

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPReplay:CREatelist
```

class CreateList
CreateList commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class GetStruct
Response structure. Fields:

- Files: List[str]: No parameter help available
- Iterations: List[int]: No parameter help available
- Interfaces: List[enums.NetworkInterface]: No parameter help available

get() → GetStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:CREatelist
value: GetStruct = driver.configure.data.measurement.ipReplay.createList.get()
```

Adds a single file to the playlist (measurement must be OFF) . A query returns all files in the playlist as follows: {<FileName>, <Iteration>, <NetworkInterface>}file 1, {...}file 2, ..., {...}file n To query a list of all files in the ip_replay directory, see method RsCmwDau.Data.Measurement.IpReplay.FileList.fetch.

return structure: for return value, see the help for GetStruct structure arguments.

set(*filename: str*, *iteration: Optional[float] = None*, *network_interface: Optional[RsCmwDau.enums.NetworkInterface] = None*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:CREatelist
driver.configure.data.measurement.ipReplay.createList.set(filename = '1',␣
↪iteration = 1.0, network_interface = enums.NetworkInterface.IP)
```

Adds a single file to the playlist (measurement must be OFF) . A query returns all files in the playlist as follows: {<FileName>, <Iteration>, <NetworkInterface>}file 1, {...}file 2, ..., {...}file n To query a list of all files in the ip_replay directory, see method RsCmwDau.Data.Measurement.IpReplay.FileList.fetch.

param filename File name as a string. Specify the file name with extension but without path, for example 'myfile.pcap'.

param iteration Specifies how often the file is replayed Range: 0 to 10E+3

param network_interface LANDau | IP | MULTicast LANDau: IP traffic to the LAN DAU connector IP: IP unicast traffic to the DUT MULTicast: IP multicast traffic to the DUT

7.2.1.2.10.2 RemoveList

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPReplay:REMovelist
```

class RemoveList

RemoveList commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:REMovelist
driver.configure.data.measurement.ipReplay.removeList.set()
```

Removes all files from the playlist (measurement must be OFF) .

set_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:REMovelist
driver.configure.data.measurement.ipReplay.removeList.set_with_opc()
```

Removes all files from the playlist (measurement must be OFF) .

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.2.10.3 Iteration

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPReplay:ITERation
```

class Iteration

Iteration commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class GetStruct

Response structure. Fields:

- Files_Names: List[str]: No parameter help available
- Iterations: List[int]: No parameter help available

get() → GetStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:ITERation
value: GetStruct = driver.configure.data.measurement.ipReplay.iteration.get()
```

Specifies how often a selected file in the playlist is replayed. A query returns all files in the playlist as follows: {<FileName>, <Iteration>}file 1, {...}file 2, ..., {...}file n

return structure: for return value, see the help for GetStruct structure arguments.

set(*filename: str*, *iteration: int*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:ITERation
driver.configure.data.measurement.ipReplay.iteration.set(filename = '1',␣
↪iteration = 1)
```

Specifies how often a selected file in the playlist is replayed. A query returns all files in the playlist as follows: {<FileName>, <Iteration>}file 1, {...}file 2, ..., {...}file n

param filename File name as a string. Specify the file name with extension but without path, for example 'myfile.pcap'.

param iteration Specifies how often the file is replayed Range: 0 to 10E+3

7.2.1.2.10.4 Interface

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPReplay:INTerface
```

class Interface

Interface commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class GetStruct

Response structure. Fields:

- Files_Names: List[str]: No parameter help available
- Network_Interfaces: List[enums.NetworkInterface]: No parameter help available

get() → GetStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:INTerface
value: GetStruct = driver.configure.data.measurement.ipReplay.interface.get()
```

Specifies the network interface for a selected file in the playlist. A query returns all files in the playlist as follows: {<FileName>, <NetworkInterface>}file 1, {...}file 2, ..., {...}file n

return structure: for return value, see the help for GetStruct structure arguments.

set(*filename: str*, *network_interface: RsCmwDau.enums.NetworkInterface*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:INTerface
driver.configure.data.measurement.ipReplay.interface.set(filename = '1',␣
↪network_interface = enums.NetworkInterface.IP)
```

Specifies the network interface for a selected file in the playlist. A query returns all files in the playlist as follows: {<FileName>, <NetworkInterface>}file 1, {...}file 2, ..., {...}file n

param filename File name as a string. Specify the file name with extension but without path, for example 'myfile.pcap'.

param network_interface LANDau | IP | MULTicast LANDau: IP traffic to the LAN DAU connector IP: IP unicast traffic to the DUT MULTicast: IP multicast traffic to the DUT

7.2.1.2.10.5 PlayAll

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPReplay:PLAYall
```

class PlayAll

PlayAll commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:PLAYall
driver.configure.data.measurement.ipReplay.playAll.set()
```

Starts replaying the playlist.

set_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:PLAYall
driver.configure.data.measurement.ipReplay.playAll.set_with_opc()
```

Starts replaying the playlist.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.2.10.6 StopAll

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:IPReplay:STOPall
```

class StopAll

StopAll commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:STOPall
driver.configure.data.measurement.ipReplay.stopAll.set()
```

Stops replaying the playlist.

set_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:IPReplay:STOPall
driver.configure.data.measurement.ipReplay.stopAll.set_with_opc()
```

Stops replaying the playlist.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.2.11 Nimpairments<Impairments>

RepCap Settings

```
# Range: Ix1 .. Ix15
rc = driver.configure.data.measurement.nimpairments.repcap_impairments_get()
driver.configure.data.measurement.nimpairments.repcap_impairments_set(repcap.Impairments.
↪Ix1)
```

class Nimpairments

Nimpairments commands group definition. 10 total commands, 10 Sub-groups, 0 group commands Repeated Capability: Impairments, default value after init: Impairments.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.nimpairments.clone()
```

Subgroups

7.2.1.2.11.1 Enable

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:ENABle
```

class Enable

Enable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments*=<*Impairments.Default: -1*>) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:ENABle
value: bool = driver.configure.data.measurement.nimpairments.enable.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return enable: No help available

set(*enable: bool*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:ENABle
driver.configure.data.measurement.nimpairments.enable.set(enable = False,␣
↪impairments = repcap.Impairments.Default)
```

No command help available

param enable No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.2 IpAddress

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:IPADdress
```

class IpAddress

IpAddress commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:IPADdress
value: str = driver.configure.data.measurement.nimpairments.ipAddress.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return ip_address: No help available

set(*ip_address: str*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:IPADdress
driver.configure.data.measurement.nimpairments.ipAddress.set(ip_address = '1',␣
↪impairments = repcap.Impairments.Default)
```

No command help available

param ip_address No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.3 Prange

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:PRANge
```

class Prange

Prange commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class PrangeStruct

Structure for setting input parameters. Fields:

- Start_Port: int: No parameter help available
- End_Port: int: No parameter help available

get(*impairments=<Impairments.Default: -1>*) → PrangeStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:PRANge
value: PrangeStruct = driver.configure.data.measurement.nimpairments.prange.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return structure: for return value, see the help for PrangeStruct structure arguments.

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Measurement_.Nimpairments_.Prange.Prange.PrangeStruct*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:PRANge
driver.configure.data.measurement.nimpairments.prange.set(value = [PROPERTY_
↪STRUCT_NAME](), impairments = repcap.Impairments.Default)
```

No command help available

param structure for set value, see the help for PrangeStruct structure arguments.

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.4 PlRate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:PLRate
```

class PlRate

PlRate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:PLRate
value: float = driver.configure.data.measurement.nimpairments.plRate.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return packet_loss_rate: No help available

set(*packet_loss_rate: float*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:PLRate
driver.configure.data.measurement.nimpairments.plRate.set(packet_loss_rate = 1.
↪0, impairments = repcap.Impairments.Default)
```

No command help available

param packet_loss_rate No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.5 Jitter

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:JITTer
```

class Jitter

Jitter commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:JITTer
value: float = driver.configure.data.measurement.nimpairments.jitter.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return jitter: No help available

set(*jitter: float*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:JITTer
driver.configure.data.measurement.nimpairments.jitter.set(jitter = 1.0,␣
↪impairments = repcap.Impairments.Default)
```

No command help available

param jitter No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.6 JitterDistribution

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:JDIStribut
```

class JitterDistribution

JitterDistribution commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → RsCmwDau.enums.JitterDistrib

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:JDIStribut
value: enums.JitterDistrib = driver.configure.data.measurement.nimpairments.
↪jitterDistribution.get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return jitter_distr: No help available

set(*jitter_distr: RsCmwDau.enums.JitterDistrib*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:JDIStribut
driver.configure.data.measurement.nimpairments.jitterDistribution.set(jitter_
↪distr = enums.JitterDistrib.NORMal, impairments = repcap.Impairments.Default)
```

No command help available

param jitter_distr No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.7 Delay

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:DELay
```

class Delay

Delay commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:DELay
value: float = driver.configure.data.measurement.nimpairments.delay.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return delay: No help available

set(*delay: float*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:DELay
driver.configure.data.measurement.nimpairments.delay.set(delay = 1.0,␣
↪impairments = repcap.Impairments.Default)
```

No command help available

param delay No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.8 Crate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:CRATe
```

class Crate

Crate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:CRATe
value: float = driver.configure.data.measurement.nimpairments.crate.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return corrupt_rate: No help available

set(*corrupt_rate: float*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:CRATe
driver.configure.data.measurement.nimpairments.crate.set(corrupt_rate = 1.0,␣
↪impairments = repcap.Impairments.Default)
```

No command help available

param corrupt_rate No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.9 Drate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:DRATe
```

class Drate

Drate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:DRATe
value: float = driver.configure.data.measurement.nimpairments.drate.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return duplicate_rate: No help available

set(*duplicate_rate: float*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:DRATe
driver.configure.data.measurement.nimpairments.drate.set(duplicate_rate = 1.0,␣
↪impairments = repcap.Impairments.Default)
```

No command help available

param duplicate_rate No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.11.10 Rrate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:NIMPairments<Impairments>:RRATe
```

class Rrate

Rrate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*impairments=<Impairments.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:RRATe
value: float = driver.configure.data.measurement.nimpairments.rrate.
↪get(impairments = repcap.Impairments.Default)
```

No command help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

return reorder_rate: No help available

set(*reorder_rate: float*, *impairments=<Impairments.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:NIMPairments<Index>:RRATe
driver.configure.data.measurement.nimpairments.rrate.set(reorder_rate = 1.0,␣
↪impairments = repcap.Impairments.Default)
```

No command help available

param reorder_rate No help available

param impairments optional repeated capability selector. Default value: Ix1 (settable in the interface 'Nimpairments')

7.2.1.2.12 Qos

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:MODE
```

class Qos

Qos commands group definition. 18 total commands, 1 Sub-groups, 1 group commands

get_mode() → RsCmwDau.enums.QosMode

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:MODE
value: enums.QosMode = driver.configure.data.measurement.qos.get_mode()
```

No command help available

return mode: No help available

set_mode(*mode: RsCmwDau.enums.QosMode*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:MODE
driver.configure.data.measurement.qos.set_mode(mode = enums.QosMode.PRIO)
```

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.data.measurement.qos.clone()
```

Subgroups

7.2.1.2.12.1 FilterPy<Fltr>

RepCap Settings

```
# Range: Ix1 .. Ix15
rc = driver.configure.data.measurement.qos.filterPy.repcap_fltr_get()
driver.configure.data.measurement.qos.filterPy.repcap_fltr_set(repcap.Fltr.Ix1)
```

class FilterPy

FilterPy commands group definition. 17 total commands, 17 Sub-groups, 0 group commands Repeated Capability: Fltr, default value after init: Fltr.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.qos.filterPy.clone()
```

Subgroups

7.2.1.2.12.2 Remove

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:REMove
```

class Remove

Remove commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set(*fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:REMove
driver.configure.data.measurement.qos.filterPy.remove.set(fltr = repcap.Fltr.
↪Default)
```

Deletes the QoS profile number <Index>.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

set_with_opc(*fltr=<Fltr.Default: -1>*) → None

7.2.1.2.12.3 HopLimit

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:HOPLmt
```

class HopLimit

HopLimit commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:HOPLmt
value: int = driver.configure.data.measurement.qos.filterPy.hopLimit.get(fltr =␣
↪repcap.Fltr.Default)
```

Sets the hop limit of packets matching the filter criteria. The setting 0 means that the hop limit of the packets is left unchanged.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return hop_limit: Range: 0 to 255

set(*hop_limit: int*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:HOPLmt
driver.configure.data.measurement.qos.filterPy.hopLimit.set(hop_limit = 1, fltr␣
↪= repcap.Fltr.Default)
```

Sets the hop limit of packets matching the filter criteria. The setting 0 means that the hop limit of the packets is left unchanged.

param hop_limit Range: 0 to 255

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.4 Add

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer:ADD
```

class Add

Add commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer:ADD
driver.configure.data.measurement.qos.filterPy.add.set()
```

Creates a QoS profile.

set_with_opc() → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer:ADD
driver.configure.data.measurement.qos.filterPy.add.set_with_opc()
```

Creates a QoS profile.

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.2.1.2.12.5 Bitrate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:BITRate
```

class Bitrate

Bitrate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → int

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:BITRate
value: int or bool = driver.configure.data.measurement.qos.filterPy.bitrate.
↪get(fltr = repcap.Fltr.Default)
```

Specifies the maximum bit rate for a QoS profile.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return qos_bitrate: Range: 0 bit/s to 100E+9 bit/s, Unit: bit/s Additional values: OFF | ON (disables | enables the bit-rate limitation)

set(*qos_bitrate: int*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:BITRate
driver.configure.data.measurement.qos.filterPy.bitrate.set(qos_bitrate = 1,␣
↪fltr = repcap.Fltr.Default)
```

Specifies the maximum bit rate for a QoS profile.

param qos_bitrate Range: 0 bit/s to 100E+9 bit/s, Unit: bit/s Additional values: OFF | ON (disables | enables the bit-rate limitation)

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.6 SrcpRange

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:SRCPrange
```

class SrcpRange

SrcpRange commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class SrcpRangeStruct

Structure for setting input parameters. Fields:

- Start_Port: int: Range: 0 to 65535
- End_Port: int: Range: 0 to 65535

get(*fltr=<Fltr.Default: -1>*) → SrcpRangeStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:SRCPrange
value: SrcpRangeStruct = driver.configure.data.measurement.qos.filterPy.
↪srcpRange.get(fltr = repcap.Fltr.Default)
```

Specifies a source port range as filter criterion for IP packets. To disable source port filtering, set both values to zero.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return structure: for return value, see the help for SrcpRangeStruct structure arguments.

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Measurement_.Qos_.FilterPy_.SrcpRange.SrcpRange.SrcpRangeStruct*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:SRCPrange
driver.configure.data.measurement.qos.filterPy.srcpRange.set(value = [PROPERTY_
↪STRUCT_NAME](), fltr = repcap.Fltr.Default)
```

Specifies a source port range as filter criterion for IP packets. To disable source port filtering, set both values to zero.

param structure for set value, see the help for SrcpRangeStruct structure arguments.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.7 Protocol

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:PROTocol
```

class Protocol

Protocol commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → RsCmwDau.enums.ProtocolB

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:PROTocol
value: enums.ProtocolB = driver.configure.data.measurement.qos.filterPy.
↪protocol.get(fltr = repcap.Fltr.Default)
```

Specifies the protocol as filter criterion for IP packets.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return protocol: ALL | TCP | UDP No filtering, TCP only, UDP only

set(*protocol: RsCmwDau.enums.ProtocolB*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:PROTocol
driver.configure.data.measurement.qos.filterPy.protocol.set(protocol = enums.
↪ProtocolB.ALL, fltr = repcap.Fltr.Default)
```

Specifies the protocol as filter criterion for IP packets.

param protocol ALL | TCP | UDP No filtering, TCP only, UDP only

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.8 TcpAckPrio

class TcpAckPrio

TcpAckPrio commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.data.measurement.qos.filterPy.tcpAckPrio.clone()
```

Subgroups

7.2.1.2.12.9 Enable

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:TCPackprio:ENABle
```

class Enable

Enable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr*=<*Fltr.Default: -1*>) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:TCPackprio:ENABle
value: bool = driver.configure.data.measurement.qos.filterPy.tcpAckPrio.enable.
↪get(fltr = repcap.Fltr.Default)
```

Enables the prioritization of TCP acknowledgments over other packets.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return enable: OFF | ON

set(*enable: bool*, *fltr*=<*Fltr.Default: -1*>) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:TCPackprio:ENABle
driver.configure.data.measurement.qos.filterPy.tcpAckPrio.enable.set(enable =␣
↪False, fltr = repcap.Fltr.Default)
```

Enables the prioritization of TCP acknowledgments over other packets.

param enable OFF | ON

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.10 Enable

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:ENABle
```

class Enable

Enable commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr*=<*Fltr.Default: -1*>) → bool

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:ENABle
value: bool = driver.configure.data.measurement.qos.filterPy.enable.get(fltr =␣
↪repcap.Fltr.Default)
```

Enables or disables a QoS profile. To use QoS profiles, you must also activate the QoS feature, see method RsCmwDau. Source.Data.Measurement.Qos.State.set.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return enable: OFF | ON

set(*enable: bool*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:ENABle
driver.configure.data.measurement.qos.filterPy.enable.set(enable = False, fltr␣
↪= repcap.Fltr.Default)
```

Enables or disables a QoS profile. To use QoS profiles, you must also activate the QoS feature, see method RsCmwDau. Source.Data.Measurement.Qos.State.set.

param enable OFF | ON

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.11 IpAddress

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:IPADdress
```

class IpAddress

IpAddress commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → str

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:IPADdress
value: str = driver.configure.data.measurement.qos.filterPy.ipAddress.get(fltr␣
↪= repcap.Fltr.Default)
```

Specifies the destination address as filter criterion for IP packets.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return ip_address: String indicating a full IPv4 address or a full IPv6 address or an IPv6 prefix

set(*ip_address: str*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:IPADdress
driver.configure.data.measurement.qos.filterPy.ipAddress.set(ip_address = '1',␣
↪fltr = repcap.Fltr.Default)
```

Specifies the destination address as filter criterion for IP packets.

param ip_address String indicating a full IPv4 address or a full IPv6 address or an IPv6 prefix

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.12 Prange

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:PRANge
```

class Prange

Prange commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class PrangeStruct

Structure for setting input parameters. Fields:

- Start_Port: int: Range: 0 to 65535
- End_Port: int: Range: 0 to 65535

get(*fltr=<Fltr.Default: -1>*) → PrangeStruct

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:PRANge
value: PrangeStruct = driver.configure.data.measurement.qos.filterPy.prange.
↪get(fltr = repcap.Fltr.Default)
```

Specifies a destination port range as filter criterion for IP packets. To disable destination port filtering, set both values to zero.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return structure: for return value, see the help for PrangeStruct structure arguments.

set(*structure: RsCmwDau.Implementations.Configure_.Data_.Measurement_.Qos_.FilterPy_.Prange.Prange.PrangeStruct*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:PRANge
driver.configure.data.measurement.qos.filterPy.prange.set(value = [PROPERTY_
↪STRUCT_NAME](), fltr = repcap.Fltr.Default)
```

Specifies a destination port range as filter criterion for IP packets. To disable destination port filtering, set both values to zero.

param structure for set value, see the help for PrangeStruct structure arguments.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.13 PlRate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:PLRate
```

class PlRate

PlRate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:PLRate
value: float = driver.configure.data.measurement.qos.filterPy.plRate.get(fltr =␣
↪repcap.Fltr.Default)
```

Specifies the packet loss rate for a QoS profile.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return packet_loss_rate: Range: 0 % to 100 %, Unit: %

set(*packet_loss_rate: float*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:PLRate
driver.configure.data.measurement.qos.filterPy.plRate.set(packet_loss_rate = 1.
↪0, fltr = repcap.Fltr.Default)
```

Specifies the packet loss rate for a QoS profile.

param packet_loss_rate Range: 0 % to 100 %, Unit: %

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.14 Jitter

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:JITTer
```

class Jitter

Jitter commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:JITTer
value: float = driver.configure.data.measurement.qos.filterPy.jitter.get(fltr =␣
↪repcap.Fltr.Default)
```

Specifies the jitter for a QoS profile. The jitter must be smaller than or equal to the configured delay.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return jitter: Range: 0 s to 10 s, Unit: s

set(*jitter: float*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:JITTer
driver.configure.data.measurement.qos.filterPy.jitter.set(jitter = 1.0, fltr =␣
↪repcap.Fltr.Default)
```

Specifies the jitter for a QoS profile. The jitter must be smaller than or equal to the configured delay.

param jitter Range: 0 s to 10 s, Unit: s

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.15 JitterDistribution

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:JDIStribut
```

class JitterDistribution

JitterDistribution commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → RsCmwDau.enums.JitterDistrib

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:JDIStribut
value: enums.JitterDistrib = driver.configure.data.measurement.qos.filterPy.
↪jitterDistribution.get(fltr = repcap.Fltr.Default)
```

Specifies the jitter distribution for a QoS profile.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return jitter_distr: UNIForm | NORMal | PAReto | PNORmal Uniform, normal, pareto, pareto normal

set(*jitter_distr: RsCmwDau.enums.JitterDistrib*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:JDIStribut
driver.configure.data.measurement.qos.filterPy.jitterDistribution.set(jitter_
↪distr = enums.JitterDistrib.NORMal, fltr = repcap.Fltr.Default)
```

Specifies the jitter distribution for a QoS profile.

param jitter_distr UNIForm | NORMal | PAReto | PNORmal Uniform, normal, pareto, pareto normal

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.16 Delay

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:DELay
```

class Delay

Delay commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:DELay
value: float = driver.configure.data.measurement.qos.filterPy.delay.get(fltr =␣
↪repcap.Fltr.Default)
```

Specifies the delay for a QoS profile.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return delay: Range: 0 s to 10 s, Unit: s

set(*delay: float*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:DELay
driver.configure.data.measurement.qos.filterPy.delay.set(delay = 1.0, fltr =␣
↪repcap.Fltr.Default)
```

Specifies the delay for a QoS profile.

param delay Range: 0 s to 10 s, Unit: s

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.17 Crate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:CRATe
```

class Crate

Crate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:CRATe
value: float = driver.configure.data.measurement.qos.filterPy.crate.get(fltr =␣
↪repcap.Fltr.Default)
```

Specifies the percentage of packets to be corrupted.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return corrupt_rate: Range: 0 % to 100 %, Unit: %

set(*corrupt_rate: float*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:CRATe
driver.configure.data.measurement.qos.filterPy.crate.set(corrupt_rate = 1.0,␣
↪fltr = repcap.Fltr.Default)
```

Specifies the percentage of packets to be corrupted.

param corrupt_rate Range: 0 % to 100 %, Unit: %

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.18 Drate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:DRATe
```

class Drate

Drate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:DRATe
value: float = driver.configure.data.measurement.qos.filterPy.drate.get(fltr =␣
↪repcap.Fltr.Default)
```

Specifies the percentage of packets to be duplicated.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return duplicate_rate: Range: 0 % to 100 %, Unit: %

set(*duplicate_rate: float*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:DRATe
driver.configure.data.measurement.qos.filterPy.drate.set(duplicate_rate = 1.0,␣
↪fltr = repcap.Fltr.Default)
```

Specifies the percentage of packets to be duplicated.

param duplicate_rate Range: 0 % to 100 %, Unit: %

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.1.2.12.19 Rrate

SCPI Commands

```
CONFigure:DATA:MEASurement<MeasInstance>:QOS:FILTer<Fltr>:RRATe
```

class Rrate

Rrate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*fltr=<Fltr.Default: -1>*) → float

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:RRATe
value: float = driver.configure.data.measurement.qos.filterPy.rrate.get(fltr =␣
↪repcap.Fltr.Default)
```

Specifies the reordering rate for a QoS profile.

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

return reorder_rate: Range: 0 % to 100 %, Unit: %

set(*reorder_rate: float*, *fltr=<Fltr.Default: -1>*) → None

```
# SCPI: CONFigure:DATA:MEASurement<Instance>:QOS:FILTer<Index>:RRATe
driver.configure.data.measurement.qos.filterPy.rrate.set(reorder_rate = 1.0,␣
↪fltr = repcap.Fltr.Default)
```

Specifies the reordering rate for a QoS profile.

param reorder_rate Range: 0 % to 100 %, Unit: %

param fltr optional repeated capability selector. Default value: Ix1 (settable in the interface 'FilterPy')

7.2.2 Switch

class Switch

Switch commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.switch.clone()
```

Subgroups

7.2.2.1 To

class To
To commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.configure.switch.to.clone()
```

Subgroups

7.2.2.1.1 Dac

SCPI Commands

```
CONFigure:SWITch:TO:DAC
```

class Dac
Dac commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

set() → None

```
# SCPI: CONFigure:SWITch:TO:DAC
driver.configure.switch.to.dac.set()
```

No command help available

set_with_opc() → None

```
# SCPI: CONFigure:SWITch:TO:DAC
driver.configure.switch.to.dac.set_with_opc()
```

No command help available

Same as set, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

7.3 Source

class Source
Source commands group definition. 14 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.clone()
```

Subgroups

7.3.1 Data

class Data
Data commands group definition. 14 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.clone()
```

Subgroups

7.3.1.1 Control

class Control
Control commands group definition. 12 total commands, 8 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.clone()
```

Subgroups

7.3.1.1.1 Udp

class Udp
Udp commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.udp.clone()
```

Subgroups

7.3.1.1.1.1 State

SCPI Commands

```
SOURce:DATA:CONTrol:UDP:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:UDP:STATe
value: enums.DauState = driver.source.data.control.udp.state.get()
```

No command help available

return state: No help available

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:CONTrol:UDP:STATe
driver.source.data.control.udp.state.set(control = False)
```

No command help available

param control No help available

7.3.1.1.2 Supl

SCPI Commands

```
SOURce:DATA:CONTrol:SUPL:REX
```

class Supl

Supl commands group definition. 4 total commands, 2 Sub-groups, 1 group commands

class RexStruct

Structure for reading output parameters. Fields:

- Reliability: int: No parameter help available
- Reliability_Msg: str: No parameter help available
- Reliability_Add_Info: str: No parameter help available

get_rex() → RexStruct

```
# SCPI: SOURce:DATA:CONTrol:SUPL:REX
value: RexStruct = driver.source.data.control.supl.get_rex()
```

No command help available

return structure: for return value, see the help for RexStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.supl.clone()
```

Subgroups

7.3.1.1.2.1 Reliability

SCPI Commands

```
SOURce:DATA:CONTrol:SUPL:RELiability:ALL
SOURce:DATA:CONTrol:SUPL:RELiability
```

class Reliability

Reliability commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class AllStruct

Structure for reading output parameters. Fields:

- Reliability: int: No parameter help available
- Reliability_Msg: str: No parameter help available
- Reliability_Add_Info: str: No parameter help available

class GetStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Reliability_Msg: str: No parameter help available
- Reliability_Add_Info: str: No parameter help available

get(*details: Optional[str] = None*) → GetStruct

```
# SCPI: SOURce:DATA:CONTrol:SUPL:RELiability
value: GetStruct = driver.source.data.control.supl.reliability.get(details = '1
↪')
```

No command help available

param details No help available

return structure: for return value, see the help for GetStruct structure arguments.

get_all() → AllStruct

```
# SCPI: SOURce:DATA:CONTrol:SUPL:RELiability:ALL
value: AllStruct = driver.source.data.control.supl.reliability.get_all()
```

No command help available

return structure: for return value, see the help for AllStruct structure arguments.

7.3.1.1.2.2 State

SCPI Commands

```
SOURce:DATA:CONTrol:SUPL:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:SUPL:STATe
value: enums.DauState = driver.source.data.control.supl.state.get()
```

No command help available

return state: No help available

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:CONTrol:SUPL:STATe
driver.source.data.control.supl.state.set(control = False)
```

No command help available

param control No help available

7.3.1.1.3 State

SCPI Commands

```
SOURce:DATA:CONTrol:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:STATe
value: enums.DauState = driver.source.data.control.state.get()
```

Switches the DAU on or off. These actions are irrelevant for normal operation of the DAU. For troubleshooting, a reboot of the DAU can be initiated by switching if off and on again.

return dau_state: OFF | PENDing | ON OFF: DAU switched off PEND: DAU has been switched on and is booting ON: DAU switched on and ready for operation

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:CONTrol:STATe
driver.source.data.control.state.set(control = False)
```

Switches the DAU on or off. These actions are irrelevant for normal operation of the DAU. For troubleshooting, a reboot of the DAU can be initiated by switching if off and on again.

param control ON | OFF Switch DAU ON or OFF

7.3.1.1.4 Dns

class Dns

Dns commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.dns.clone()
```

Subgroups

7.3.1.1.4.1 State

SCPI Commands

```
SOURce:DATA:CONTrol:DNS:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:DNS:STATe
value: enums.DauState = driver.source.data.control.dns.state.get()
```

Starts or stops the local DNS server.

return state: OFF | ON | PENDing OFF: service switched off ON: service switched on PEND: service activation or deactivation ongoing

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:CONTrol:DNS:STATe
driver.source.data.control.dns.state.set(control = False)
```

Starts or stops the local DNS server.

param control ON | OFF Switch the service ON or OFF

7.3.1.1.5 Ftp

class Ftp
Ftp commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.ftp.clone()
```

Subgroups

7.3.1.1.5.1 State

SCPI Commands

```
SOURce:DATA:CONTrol:FTP:STATe
```

class State
State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:FTP:STATe
value: enums.DauState = driver.source.data.control.ftp.state.get()
```

Starts or stops the FTP service.

return state: OFF | ON | PENDing OFF: service switched off ON: service switched on PEND: service activation or deactivation ongoing

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:CONTrol:FTP:STATe
driver.source.data.control.ftp.state.set(control = False)
```

Starts or stops the FTP service.

param control ON | OFF Switch the service ON or OFF

7.3.1.1.6 Http

SCPI Commands

```
SOURce:DATA:CONTrol:HTTP:RELiability
```

class Http
Http commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

class ReliabilityStruct

Structure for reading output parameters. Fields:

- Reliability: int: No parameter help available
- Reliability_Msg: str: No parameter help available
- Reliability_Add_Info: str: No parameter help available

get_reliability() → ReliabilityStruct

```
# SCPI: SOURce:DATA:CONTrol:HTTP:RELiability
value: ReliabilityStruct = driver.source.data.control.http.get_reliability()
```

No command help available

return structure: for return value, see the help for ReliabilityStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.http.clone()
```

Subgroups

7.3.1.1.6.1 State

SCPI Commands

```
SOURce:DATA:CONTrol:HTTP:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:HTTP:STATe
value: enums.DauState = driver.source.data.control.http.state.get()
```

Starts or stops the HTTP service.

return state: OFF | ON | PENDing OFF: service switched off ON: service switched on PEND: service activation or deactivation ongoing

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:CONTrol:HTTP:STATe
driver.source.data.control.http.state.set(control = False)
```

Starts or stops the HTTP service.

param control ON | OFF Switch the service ON or OFF

7.3.1.1.7 Ims<Ims>

RepCap Settings

```
# Range: Ix1 .. Ix2
rc = driver.source.data.control.ims.repcap_ims_get()
driver.source.data.control.ims.repcap_ims_set(repcap.Ims.Ix1)
```

class Ims

Ims commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Ims, default value after init: Ims.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.ims.clone()
```

Subgroups

7.3.1.1.7.1 State

SCPI Commands

```
SOURce:DATA:CONTrol:IMS<Ims>:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get(*ims=<Ims.Default: -1>*) → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:IMS<Suffix>:STATe
value: enums.DauState = driver.source.data.control.ims.state.get(ims = repcap.
↪Ims.Default)
```

Starts or stops the IMS service and the IMS server.

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return state: OFF | ON | PENDing OFF: service switched off ON: service switched on PEND: service activation or deactivation ongoing

set(*control: bool*, *ims=<Ims.Default: -1>*) → None

```
# SCPI: SOURce:DATA:CONTrol:IMS<Suffix>:STATe
driver.source.data.control.ims.state.set(control = False, ims = repcap.Ims.
↪Default)
```

Starts or stops the IMS service and the IMS server.

param control ON | OFF Switch the service ON or OFF

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

7.3.1.1.8 Epdg

class Epdg

Epdg commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.control.epdg.clone()
```

Subgroups

7.3.1.1.8.1 State

SCPI Commands

```
SOURce:DATA:CONTrol:EPDG:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:CONTrol:EPDG:STATe
value: enums.DauState = driver.source.data.control.epdg.state.get()
```

Starts or stops the ePDG service.

return state: OFF | ON | PENDing OFF: service switched off ON: service switched on PEND: service activation or deactivation ongoing

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:CONTrol:EPDG:STATe
driver.source.data.control.epdg.state.set(control = False)
```

Starts or stops the ePDG service.

param control ON | OFF Switch the service ON or OFF

7.3.1.2 Measurement

class Measurement
Measurement commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.measurement.clone()
```

Subgroups

7.3.1.2.1 Qos

class Qos
Qos commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.source.data.measurement.qos.clone()
```

Subgroups

7.3.1.2.1.1 FilterPy

SCPI Commands

```
SOURce:DATA:MEASurement<MeasInstance>:QOS:FILTer:CATalog
```

class FilterPy
FilterPy commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get_catalog() → str

```
# SCPI: SOURce:DATA:MEASurement<Instance>:QOS:FILTer:CATalog
value: str = driver.source.data.measurement.qos.filterPy.get_catalog()
```

Queries a list of all existing QoS profiles.

return catalog: String with comma-separated list of profile names, for example: 'Filter 1,Filter 2,Filter 3'

7.3.1.2.1.2 State

SCPI Commands

```
SOURce:DATA:MEASurement<MeasInstance>:QOS:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

get() → RsCmwDau.enums.DauState

```
# SCPI: SOURce:DATA:MEASurement<Instance>:QOS:STATe
value: enums.DauState = driver.source.data.measurement.qos.state.get()
```

Switches the QoS feature on or off. To enable QoS profiles, see method RsCmwDau.Configure.Data.Measurement.Qos.FilterPy.Enable.set.

return ni_state: OFF | PENDing | ON OFF: QoS feature switched off PEND: switching on/off is ongoing ON: QoS feature switched on

set(*control: bool*) → None

```
# SCPI: SOURce:DATA:MEASurement<Instance>:QOS:STATe
driver.source.data.measurement.qos.state.set(control = False)
```

Switches the QoS feature on or off. To enable QoS profiles, see method RsCmwDau.Configure.Data.Measurement.Qos.FilterPy.Enable.set.

param control ON | OFF Switch the QoS feature on or off

7.4 Data

class Data

Data commands group definition. 126 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.clone()
```

Subgroups

7.4.1 Control

class Control

Control commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.control.clone()
```

Subgroups

7.4.1.1 Ims<Ims>

RepCap Settings

```
# Range: Ix1 .. Ix2
rc = driver.data.control.ims.repcap_ims_get()
driver.data.control.ims.repcap_ims_set(repcap.Ims.Ix1)
```

class Ims
Ims commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Ims, default value after init: Ims.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.control.ims.clone()
```

Subgroups

7.4.1.1.1 VirtualSubscriber

class VirtualSubscriber
VirtualSubscriber commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.control.ims.virtualSubscriber.clone()
```

Subgroups

7.4.1.1.1.1 FileList

class FileList
FileList commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.control.ims.virtualSubscriber.fileList.clone()
```

Subgroups

7.4.1.1.1.2 Pcap

SCPI Commands

```
FETCh:DATA:CONTrol:IMS<Ims>:VIRTualsub:FILelist:PCAP
```

class Pcap

Pcap commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch(*ims*=<*Ims.Default: -1*>) → List[str]

```
# SCPI: FETCh:DATA:CONTrol:IMS<Suffix>:VIRTualsub:FILelist:PCAP
value: List[str] = driver.data.control.ims.virtualSubscriber.fileList.pcap.
↪fetch(ims = repcap.Ims.Default)
```

No command help available

param ims optional repeated capability selector. Default value: Ix1 (settable in the interface 'Ims')

return files: No help available

7.4.2 Measurement

class Measurement

Measurement commands group definition. 125 total commands, 8 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.clone()
```

Subgroups

7.4.2.1 IpAnalysis

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:IPANalysis
STOP:DATA:MEASurement<MeasInstance>:IPANalysis
ABORt:DATA:MEASurement<MeasInstance>:IPANalysis
```

class IpAnalysis

IpAnalysis commands group definition. 33 total commands, 7 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPANalysis
driver.data.measurement.ipAnalysis.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPANalysis
driver.data.measurement.ipAnalysis.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPANalysis
driver.data.measurement.ipAnalysis.initiate()
```

(continues on next page)

(continued from previous page)

```
    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPANalysis
driver.data.measurement.ipAnalysis.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPANalysis
driver.data.measurement.ipAnalysis.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPANalysis
driver.data.measurement.ipAnalysis.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.clone()
```

Subgroups

7.4.2.1.1 IpcSecurity

class IpcSecurity

IpcSecurity commands group definition. 14 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ipcSecurity.clone()
```

Subgroups

7.4.2.1.1.1 Capplication

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:CAPPlication
```

class Capplication

Capplication commands group definition. 12 total commands, 3 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Application: List[str]: Application name as string
- Flow_Id: List[int]: ID of the flow used by the connection
- Source_Ip: List[str]: IP address of the DUT as string
- Local_Port: List[int]: Port number used at the DUT side
- Destination_Ip: List[str]: Destination IP address as string
- Destination_Port: List[int]: Port number of the destination
- Fqdn: List[str]: Fully qualified domain name of the destination as string
- Ran: List[str]: Used radio access network as string
- Apn: List[str]: Access point name as string
- Protocol: List[str]: Used protocol, for example SSL or HTTP, as string
- Country_Code: List[str]: Country of the destination as string (two-letter country code)
- Location: List[str]: City of the destination, as string
- Latitude: List[str]: Latitude of the destination, as string
- Longitude: List[str]: Longitude of the destination, as string
- Ul_Data: List[float]: Layer 3 UL data exchanged via the connection Unit: bytes
- Ul_Pkt: List[int]: Number of UL packets exchanged via the connection
- Dl_Data: List[float]: Layer 3 DL data exchanged via the connection Unit: bytes
- Dl_Pkt: List[int]: Number of DL packets exchanged via the connection
- Hand_Sk_Available: List[bool]: OFF | ON Handshake information available for the connection or not
- Certif_Available: List[bool]: OFF | ON Certificate information available for the connection or not

fetch(*application_name: str*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:CAPPlication
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.fetch(application_name = '1')
```

Queries information about all connections of a selected application. The results after the reliability indicator are returned per connection: <Reliability>, {<Application>, <Flowid>, ..., <HandSkAvailable>, <CertifAvailable>}1, {...}2, ...

param application_name Application name as string

return structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ipcSecurity.capplication.clone()
```

Subgroups

7.4.2.1.1.2 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:CAPPlication:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Application: List[str]: Application name as string
- Flow_Id: List[int]: ID of the flow used by the connection
- Source_Ip: List[str]: IP address of the DUT as string
- Local_Port: List[int]: Port number used at the DUT side
- Destination_Ip: List[str]: Destination IP address as string
- Destination_Port: List[int]: Port number of the destination
- Fqdn: List[str]: Fully qualified domain name of the destination as string
- Ran: List[str]: Used radio access network as string
- Apn: List[str]: Access point name as string
- Protocol: List[str]: Used protocol, for example SSL or HTTP, as string
- Country_Code: List[str]: Country of the destination as string (two-letter country code)
- Location: List[str]: City of the destination, as string
- Latitude: List[str]: Latitude of the destination, as string
- Longitude: List[str]: Longitude of the destination, as string
- Ul_Data: List[float]: Layer 3 UL data exchanged via the connection Unit: bytes
- Ul_Pkt: List[int]: Number of UL packets exchanged via the connection
- Dl_Data: List[float]: Layer 3 DL data exchanged via the connection Unit: bytes

- Dl_Pkt: List[int]: Number of DL packets exchanged via the connection
- Hand_Sk_Available: List[bool]: OFF | ON Handshake information available for the connection or not
- Certif_Available: List[bool]: OFF | ON Certificate information available for the connection or not

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:CAPPlication:ALL
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.all.fetch()
```

Queries information about all connections of all applications. The results after the reliability indicator are returned per connection: <Reliability>, {<Application>, <Flowid>, ..., <HandSkAvailable>, <CertifAvailable>}1, {...}2, ...

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.3 Certificate

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:CAPPlication:CERTificate
```

class Certificate

Certificate commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- Subject_Certificate_Name: List[str]: No parameter help available
- Subject_Organization: List[str]: No parameter help available
- Subject_Oraganizational_Unit: List[str]: No parameter help available
- Subject_Country_Name: List[str]: No parameter help available
- Public_Key_Algorithm: List[str]: No parameter help available
- Public_Key_Length: List[int]: No parameter help available
- Sign_Algo_Id: List[str]: No parameter help available
- Sign_Algo_Name: List[str]: No parameter help available
- Signature_Key_Length: List[int]: No parameter help available
- Validitynotbefore: List[str]: No parameter help available
- Validitynot_After: List[str]: No parameter help available
- Revocation_Method: List[str]: No parameter help available
- Revocation_Status: List[str]: No parameter help available
- Perform_Date_Time: List[str]: No parameter help available
- Self_Signed: List[bool]: No parameter help available
- Trust_Store: List[str]: No parameter help available

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:CERTificate
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.certificate.fetch(flow_id = 1)
```

No command help available

param flow_id No help available

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.4 Handshake

class Handshake

Handshake commands group definition. 9 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ipcSecurity.capplication.handshake.clone()
```

Subgroups

7.4.2.1.1.5 Negotiated

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:NEGotiated
```

class Negotiated

Negotiated commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Version_Id: str: Protocol version ID as hexadecimal value
- Version_Name: str: Protocol version as string
- Cipher_Suite_Id: str: Cipher suite ID as hexadecimal value
- Cipher_Suite_Name: str: Cipher suite name as string
- Compression_Id: str: Compression method ID as hexadecimal value
- Compression_Name: str: Compression method name as string
- Register_Type: enums.RegisterType: IANA | OID Type of the register used for the signature hash algorithm pair If the value is IANA, the fields SignatureAlgoHashID and SignaHashAlgoSignID are filled with the ID values. If the value is OID, the field OID is filled with a single combined ID value for the signature hash algorithm pair. In both cases, the fields SignatureAlgoHashName and SignaHashAlgoSignName are filled with the names as strings.

- Oid: str: Signature hash algorithm ID as string
- Sign_Alg_Hash_Id: str: Hash algorithm ID as hexadecimal value
- Sign_Alg_Hash_Name: str: Hash algorithm name as string
- Sign_Alg_Sign_Id: str: Signature algorithm ID as hexadecimal value
- Sign_Alg_Sign_Name: str: Signature algorithm name as string
- Ecurve_Id: str: Elliptic curve ID as hexadecimal value
- Ec_Name: str: Elliptic curve name as string
- Ec_Type_Id: str: Elliptic curve type ID as hexadecimal value
- Ec_Type_Name: str: Elliptic curve type name as string
- Ecp_Format_Id: str: Elliptic curve point format ID as hexadecimal value - no longer supported
- Ecp_Format_Name: str: Elliptic curve point format name as string - no longer supported
- Sign_Length: int: Length of the server signature in bits
- Public_Length: int: Length of the public key of the server in bits

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:NEGotiated
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.negotiated.fetch(flow_id = 1)
```

Queries the negotiated handshake results for a specific connection.

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ipcSecurity.capplication.handshake.
↪negotiated.clone()
```

Subgroups

7.4.2.1.1.6 EcpFormats

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:NEGotiated:ECPFormats
```

class EcpFormats

EcpFormats commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Ecp_Format_Id: List[str]: Elliptic curve point format ID as hexadecimal value
- Ecp_Format_Name: List[str]: Elliptic curve point format name as string

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:NEGotiated:ECPFormats
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.negotiated.ecpFormats.fetch(flow_id = 1)
```

Queries information about the elliptic curve point formats negotiated for a specific connection. After the reliability indicator, two results are returned for each format: <Reliability>, {<ECPFormatID>, <ECPFormatName>}Format 1, {...}Format 2, ...

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.7 Offered

class Offered

Offered commands group definition. 7 total commands, 7 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ipcSecurity.capplication.handshake.offered.
↪clone()
```

Subgroups

7.4.2.1.1.8 SrIndication

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:SRINdication
```

class SrIndication

SrIndication commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Server_Id: List[str]: Server ID as hexadecimal value

- Server_Name: List[str]: Server name indication (SNI) as string
- Server_Type: List[str]: Type of the server name as string

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:SRINdication
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.offered.srIndication.fetch(flow_id = 1)
```

Queries information about the server that the client wants to contact, as sent by the client during the handshake for a specific connection. After the reliability indicator, three results are returned for each entry: <Reliability>, {<ServerID>, <ServerName>, <ServerType>}1, {...}2, ...

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.9 Ecurve

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:ECURve
```

class Ecurve

Ecurve commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Elliptic_Curve_Id: List[str]: Elliptic curve ID as hexadecimal value
- Elliptic_Curve_Name: List[str]: Elliptic curve name as string

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:ECURve
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.offered.ecurve.fetch(flow_id = 1)
```

Queries information about the elliptic curves offered during the handshake for a specific connection. After the reliability indicator, two results are returned for each elliptic curve: <Reliability>, {<EllipticCurveID>, <EllipticCurveName>}Curve 1, {...}Curve 2, ...

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.10 CipSuite

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:CIPSuite
```

class CipSuite

CipSuite commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Cipher_Suite_Id: List[str]: Cipher suite ID as hexadecimal value
- Cipher_Suite_Name: List[str]: Cipher suite name as string

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:CIPSuite
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.offered.cipSuite.fetch(flow_id = 1)
```

Queries information about the cipher suites offered during the handshake for a specific connection. After the reliability indicator, two results are returned for each cipher suite: <Reliability>, {<CipherSuiteID>, <CipherSuiteName>}Suite 1, {.. .}Suite 2, ...

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.11 ShAlgorithm

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:SHALgorithm
```

class ShAlgorithm

ShAlgorithm commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Algorithm_Hash_Id: List[str]: Hash algorithm ID as hexadecimal value
- Algorithm_Sign_Id: List[str]: Signature algorithm ID as hexadecimal value
- Algo_Hash_Name: List[str]: Hash algorithm name as string
- Algo_Sign_Name: List[str]: Signature algorithm name as string

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:SHALgorithm
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.offered.shAlgorithm.fetch(flow_id = 1)
```

Queries information about the hash algorithms and signature algorithms offered during the handshake for a specific connection. After the reliability indicator, four results are returned for each pair of algorithms: <Reliability>, {<AlgorithmHashID>, <AlgorithmSignID>, <AlgoHashName>, <AlgoSignName>}1, {...}2, ...

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.12 Version

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:VERSion
```

class Version

Version commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Version_String: str: Protocol version as string
- Version_Id: str: Protocol version ID as hexadecimal value

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:VERSion
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.offered.version.fetch(flow_id = 1)
```

Queries the protocol version offered during the handshake for a specific connection.

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.13 Compression

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:COMPression
```

class Compression

Compression commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Compression_Id: List[str]: Compression method ID as hexadecimal value
- Compression_Name: List[str]: Compression method name as string

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:COMPression
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.offered.compression.fetch(flow_id = 1)
```

Queries information about the compression methods offered during the handshake for a specific connection. After the reliability indicator, two results are returned for each method: <Reliability>, {<CompressionID>, <CompressionName>}Method 1, {...}Method 2, ...

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.14 EcpFormat

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:ECPFormat
```

class EcpFormat

EcpFormat commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Ecp_Format_Id: List[str]: Elliptic curve point format ID as hexadecimal value
- Ecp_Format_Name: List[str]: Elliptic curve point format name as string

fetch(*flow_id: int*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:IPCSecurity:CAPPlication:HANDshake:OFFered:ECPFormat
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.
↪capplication.handshake.offered.ecpFormat.fetch(flow_id = 1)
```

Queries information about the elliptic curve point formats offered during the handshake for a specific connection. After the reliability indicator, two results are returned for each format: <Reliability>, {<ECPFormatID>, <ECPFormatName>}Format 1, {...}Format 2, ...

param flow_id Selects the connection for which information is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.15 Kyword

class Kyword

Kyword commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ipcSecurity.kyword.clone()
```

Subgroups

7.4.2.1.1.16 Search

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:KYWord:SEARch
```

class Search

Search commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Keyword: List[str]: Found keyword as string
- Count: List[int]: How often the keyword was found
- Dst_Ip: List[str]: IP address of the destination as string
- Fqdn: List[str]: FQDN of the destination as string
- Application: List[str]: Application using the connection, as string
- Direction: List[enums.DirectionA]: DL | UL | UNKN Direction of the transmission - downlink, uplink or unknown

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:KYWord:SEARch
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.kyword.
↪search.fetch()
```

Queries the keyword search results. After the reliability indicator, six results are returned for each found keyword: <Reliability>, {<Keyword>, <Count>, <DstIP>, <FQDN>, <Application>, <Direction>}1, {...}2, ...

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.1.17 PrtScan

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:IPCSecurity:PRTScan
```

class PrtScan

PrtScan commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Port: List[int]: Port number
- Protocol: List[str]: Layer 4 protocol

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:IPCSecurity:PRTScan
value: FetchStruct = driver.data.measurement.ipAnalysis.ipcSecurity.prtScan.
↪fetch()
```

Queries the results of a port scan. After the reliability indicator, two parameters are returned for each open port: <Reliability>, {<Port>, <Protocol>}1, ..., {<Port>, <Protocol>}n If there is no open port, you get: <Reliability>, INV, INV

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.2 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:STATe
value: enums.ResourceState = driver.data.measurement.ipAnalysis.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.state.clone()
```

Subgroups

7.4.2.1.2.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available
- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:STATe:ALL
value: FetchStruct = driver.data.measurement.ipAnalysis.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.3 Dpcp

class Dpcp

Dpcp commands group definition. 4 total commands, 4 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.dpcp.clone()
```

Subgroups

7.4.2.1.3.1 DpConnection

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:DPCP:DPConnection
```

class DpConnection

DpConnection commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Rem_Destination: List[str]: IP address of the remote destination as string
- Conn_Data: List[float]: Data transported via the connection, as absolute number Unit: byte
- Conn_Percent: List[float]: Data transported via the connection, as percentage of total transported data Range: 0 % to 100 %, Unit: %

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPConnection
value: FetchStruct = driver.data.measurement.ipAnalysis.dpcp.dpConnection.
↪fetch()
```

Queries the 'Data per Connection' results. After the reliability indicator, three results are returned for each connection: <Reliability>, {<RemDestination>, <ConnData>, <ConnPercent>}conn 1, {...}conn 2, ...

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.3.2 DpProtocol

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:DPCP:DPPRotocol
```

class DpProtocol

DpProtocol commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Protocol: List[str]: Used protocol as string
- Prot_Data: List[float]: Data transported via the protocol, as absolute number Unit: byte
- Prot_Percent: List[float]: Data transported via the protocol, as percentage of total transported data Range: 0 % to 100 %, Unit: %

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPPRotocol
value: FetchStruct = driver.data.measurement.ipAnalysis.dpcp.dpProtocol.fetch()
```

Queries the 'Data per Protocol' results. After the reliability indicator, three results are returned for each protocol: <Reliability>, {<Protocol>, <ProtData>, <ProtPercent>}protocol 1, {...}protocol 2, ...

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.3.3 DpLayer

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:DPCP:DPLayer
```

class DpLayer

DpLayer commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Layer: List[str]: String with the contents of column 'Layer' (feature, application or protocol)
- Layer_Data: List[float]: Amount of transported data, as absolute value Unit: byte
- Layer_Percent: List[float]: Amount of transported data, as percentage of total transported data Range: 0 % to 100 %, Unit: %

fetch(*layer_depth: RsCmwDau.enums.Layer*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPLayer
value: FetchStruct = driver.data.measurement.ipAnalysis.dpcp.dpLayer.
↪fetch(layer_depth = enums.Layer.APP)
```

Queries the 'Data per Layer' results. After the reliability indicator, three values are returned for each result table row: <Reliability>, {<Layer>, <LayerData>, <LayerPercent>}row 1, {...}row 2, ...

param layer_depth FEATure | APP | L7 | L4 | L3 Selects the highest layer at which the packets are analyzed

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.3.4 DpApplic

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:DPCP:DPAPplic
```

class DpApplic

DpApplic commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- App: List[str]: String with the contents of column 'Application' (application or protocol)
- App_Data: List[float]: Amount of transported data, as absolute value Unit: byte
- App_Percent: List[float]: Amount of transported data, as percentage of total transported data Range: 0 % to 100 %, Unit: %

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:DPCP:DPAPplic
value: FetchStruct = driver.data.measurement.ipAnalysis.dpcp.dpApplic.fetch()
```

Queries the 'Data per Application' results of the current layer. To navigate between the layers, see method RsCmwDau. Configure.Data.Measurement.IpAnalysis.Dpcp.DpApplic.app. After the reliability indicator, three values are returned for each result table row: <Reliability>, {<App>, <AppData>, <AppPercent>}row 1, {...}row 2, ...

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.4 IpConnect

class IpConnect

IpConnect commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ipConnect.clone()
```

Subgroups

7.4.2.1.4.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:IPConnect:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Flow_Id: List[int]: Flow ID of the connection
- Conn_Status: List[enums.ConnStatus]: OPEN | CLOSed Connection status
- Lst: List[float]: Local system time, incremented every ms The clock starts when the instrument is switched on.
- Sys_Clock: List[int]: System clock in units of 10 ns When 1 ms is reached (100*10 ns) , the clock is reset to 0 and the local system time is incremented. Range: 0 to 99
- Protocol: List[str]: Layer 4 protocol as string ('TCP', 'UDP', ...)
- Dpi_Protocol: List[str]: Layer 7 protocol as string ('HTTP', 'FTP', ...)
- Ip_Addr_Source: List[str]: IP address of the connection source as string
- Ip_Port_Source: List[int]: Port number of the connection source Range: 0 to 65654
- Ip_Addr_Dest: List[str]: IP address of the connection destination as string
- Ip_Port_Dest: List[int]: Port number of the connection destination Range: 0 to 65654
- Overh_Down: List[float]: Downlink overhead as percentage of the packet Range: 0 % to 100 %, Unit: %
- Overh_Up: List[float]: Uplink overhead as percentage of the packet Range: 0 % to 100 %, Unit: %
- Avg_Ps_Down: List[float]: Average downlink packet size Range: 0 bytes to 65535 bytes, Unit: bytes
- Avg_Ps_Up: List[float]: Average uplink packet size Range: 0 bytes to 65535 bytes, Unit: bytes
- App: List[str]: Application name as string
- Country: List[str]: Country of the destination as string (two-letter country code)

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:IPConnect:ALL
value: FetchStruct = driver.data.measurement.ipAnalysis.ipConnect.all.fetch()
```

Queries the 'IP Connectivity' results for all connections. After the reliability indicator, results are returned per connection (flow) : <Reliability>, {<FlowID>, <ConnStatus>, <LST>, <SysClock>, <Protocol>, <DPIProtocol>, <IPAddrSource>, <IPPortSource>, <IPAddrDest>, <IPPortDest>, <OverhDown>, <OverhUp>, <AvgPSDown>, <AvgPSUp>, <App>, <Country>}conn 1, {... }conn 2, ...

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.5 TcpAnalysis

class TcpAnalysis

TcpAnalysis commands group definition. 5 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.tcpAnalysis.clone()
```

Subgroups

7.4.2.1.5.1 Rtt

class Rtt

Rtt commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.tcpAnalysis.rtt.clone()
```

Subgroups

7.4.2.1.5.2 Trace

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:RTT:TRACe
```

class Trace

Trace commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch(*flow_id: float*) → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:RTT:TRACe
value: List[float] = driver.data.measurement.ipAnalysis.tcpAnalysis.rtt.trace.
↪fetch(flow_id = 1.0)
```

Queries the round-trip time trace for a specific connection, selected via its flow ID. The trace values are returned from right to left (last to first measurement) .

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

param flow_id Selects the connection for which the trace is queried

return rtt: Comma-separated list of round-trip time values Range: 0 ms to 5000 ms, Unit: ms

7.4.2.1.5.3 Retransmiss

class Retransmiss

Retransmiss commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.tcpAnalysis.retransmiss.clone()
```

Subgroups

7.4.2.1.5.4 Trace

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:RETRansmiss:TRACe
```

class Trace

Trace commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Re_Tx_Ul: List[float]: Uplink retransmission rate Range: 0 % to 100 %, Unit: %
- Re_Tx_Dl: List[float]: Downlink retransmission rate Range: 0 % to 100 %, Unit: %

fetch(*flow_id: float*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:IPANalysis:TCPanalysis:RETRansmiss:TRACe
value: FetchStruct = driver.data.measurement.ipAnalysis.tcpAnalysis.retransmiss.
↪trace.fetch(flow_id = 1.0)
```

Queries the retransmission traces for a specific connection, selected via its flow ID. The values for the uplink and downlink traces are returned in pairs: <Reliability>, <ReTxUL>1, <ReTxDL>1, <ReTxUL>2, <ReTxDL>2, ... The traces are returned from right to left (last to first measurement) .

param flow_id Selects the connection for which the trace is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.5.5 Wsize

class Wsize
Wsize commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.tcpAnalysis.wsize.clone()
```

Subgroups

7.4.2.1.5.6 Trace

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:WSIZe:TRACe
```

class Trace
Trace commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct
Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Wsize_Ul: List[float]: Uplink TCP window size Unit: byte
- Wsize_Dl: List[float]: Downlink TCP window size Unit: byte

fetch(*flow_id: float*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:WSIZe:TRACe
value: FetchStruct = driver.data.measurement.ipAnalysis.tcpAnalysis.wsize.trace.
↪fetch(flow_id = 1.0)
```

Queries the window size traces for a specific connection, selected via its flow ID. The values for the uplink and downlink traces are returned in pairs: <Reliability>, <WSizeUL>1, <WSizeDL>1, <WSizeUL>2, <WSizeDL>2, ... The traces are returned from right to left (last to first measurement) .

param flow_id Selects the connection for which the trace is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.5.7 Throughput

class Throughput

Throughput commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.tcpAnalysis.throughput.clone()
```

Subgroups

7.4.2.1.5.8 Trace

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:THRoughput:TRACe
```

class Trace

Trace commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Thrpt_Ul: List[float]: Uplink throughput Range: 0 bit/s to 4E+9 bit/s, Unit: bit/s
- Thrpt_Dl: List[float]: Downlink throughput Range: 0 bit/s to 4E+9 bit/s, Unit: bit/s

fetch(*flow_id: float*) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:THRoughput:TRACe
value: FetchStruct = driver.data.measurement.ipAnalysis.tcpAnalysis.throughput.
↪trace.fetch(flow_id = 1.0)
```

Queries the throughput traces for a specific connection, selected via its flow ID. The values for the uplink and downlink traces are returned in pairs: <Reliability>, <ThrptUL>1, <ThrptDL>1, <ThrptUL>2, <ThrptDL>2, ... The traces are returned from right to left (last to first measurement) .

param flow_id Selects the connection for which the trace is queried

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.5.9 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:TCPanalysis:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Flow_Id: List[int]: Flow ID of the connection
- Th_Down: List[float]: Throughput in downlink direction Unit: bit/s
- Tcpws_Down: List[enums.OverhUp]: OK | FULL Threshold check result for downlink TCP window size
- Retr_Down: List[enums.OverhUp]: OK | NOK Threshold check result for downlink retransmissions
- Overh_Down: List[enums.OverhUp]: OK | NOK Only for backward compatibility - no longer used
- Th_Up: List[float]: Throughput in uplink direction Unit: bit/s
- Tcpws_Up: List[enums.OverhUp]: OK | FULL Threshold check result for uplink TCP window size
- Retr_Up: List[enums.OverhUp]: OK | NOK Threshold check result for uplink retransmissions
- Overh_Up: List[enums.OverhUp]: OK | NOK Only for backward compatibility - no longer used
- Destination: List[str]: Destination address as string
- Rtt_Status: List[enums.OverhUp]: OK | NOK Threshold check result for round-trip time
- Pkt_Size_Ul: List[int]: Layer 3 uplink packet size Unit: byte
- Pkt_Size_Dl: List[int]: Layer 3 downlink packet size Unit: byte

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:TCPanalysis:ALL
value: FetchStruct = driver.data.measurement.ipAnalysis.tcpAnalysis.all.fetch()
```

Queries the threshold check and throughput results for all connections. After the reliability indicator, 13 results are returned for each connection (flow) : <Reliability>, {<FlowID>, <ThDown>, <TCPWSDown>, <RetrDown>, <OverhDown>, <ThUp>, <TCPWSUp>, <RetrUp>, <OverhUp>, <Destination>, <RTTStatus>, <PKTSizeUL>, <PKTSizeDL>}connection 1, {...}connection 2, ...

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.6 FtTrigger

class FtTrigger

FtTrigger commands group definition. 3 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ftTrigger.clone()
```

Subgroups

7.4.2.1.6.1 Traces<Trace>

RepCap Settings

```
# Range: Ix1 .. Ix10
rc = driver.data.measurement.ipAnalysis.ftTrigger.traces.repcap_trace_get()
driver.data.measurement.ipAnalysis.ftTrigger.traces.repcap_trace_set(repcap.Trace.Ix1)
```

class Traces

Traces commands group definition. 1 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Trace, default value after init: Trace.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ftTrigger.traces.clone()
```

Subgroups

7.4.2.1.6.2 Fthroughput

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:FTTRigger:TRACes<Trace>:FTHRoughput
```

class Fthroughput

Fthroughput commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Flow_Id: int: Flow ID of the connection assigned to the selected trace index
- Throughput: List[float]: Comma-separated list of throughput values Unit: bit/s

fetch(*trace*=<*Trace.Default: -1*>) → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:FTTRigger:TRACes<TraceIndex>
↪:FTHRoughput
value: FetchStruct = driver.data.measurement.ipAnalysis.ftTrigger.traces.
↪fthroughput.fetch(trace = repcap.Trace.Default)
```

Queries a selected throughput trace. The trace is selected via its trace index. To assign a specific connection to a trace index, see method RsCmwDau.Configure.Data.Measurement.IpAnalysis.FtTrigger.Trace.TflowId.set. The trace is returned from right to left (last to first measurement) .

param trace optional repeated capability selector. Default value: Ix1 (settable in the interface 'Traces')

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.6.3 Trigger

class Trigger

Trigger commands group definition. 2 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.ftTrigger.trigger.clone()
```

Subgroups

7.4.2.1.6.4 Start

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:FTTRigger:TRIGger:STARt
```

class Start

Start commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Flow_Id: List[str]: Flow ID of the opened connection as string
- Time_Elapsed: List[float]: X-axis value of the 'open' event

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:FTTRigger:TRIGger:STARt
value: FetchStruct = driver.data.measurement.ipAnalysis.ftTrigger.trigger.start.
↪fetch()
```

Queries the event trigger trace for 'open' events. After the reliability indicator, two values are returned per 'open' event: <Reliability>, {<FlowID>, <TimeElapsed>}event 1, {<FlowID>, <TimeElapsed>}event 2, ... The trace is returned from right to left (last to first event) .

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.6.5 End

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:FTTRigger:TRIGger:END
```

class End

End commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Flow_Id: List[str]: Flow ID of the closed connection as string
- Time_Elapsed: List[float]: X-axis value of the 'close' event

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:FTTRigger:TRIGger:END
value: FetchStruct = driver.data.measurement.ipAnalysis.ftTrigger.trigger.end.
↪fetch()
```

Queries the event trigger trace for 'close' events. After the reliability indicator, two values are returned per 'close' event: <Reliability>, {<FlowID>, <TimeElapsed>}event 1, {<FlowID>, <TimeElapsed>}event 2, ... The trace is returned from right to left (last to first event) .

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.1.7 VoIms

class VoIms

VoIms commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipAnalysis.voIms.clone()
```

Subgroups

7.4.2.1.7.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPANalysis:VOIMs:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Con_Id: List[int]: Call ID
- Flows_Ids: List[str]: String containing a comma-separated list of the flow IDs related to the call (example '1,2' or '128')
- Type_Py: List[enums.AvTypeC]: AUDio | VIDeo | EMER | UNK Call type audio, video, emergency or unknown
- Origin: List[enums.Origin]: MT | MO | UNK MT: mobile-terminating call MO: mobile-originating call UNK: unknown
- State: List[enums.VoimState]: RING | EST | REL | HOLD | UNK RING: DUT ringing EST: call established REL: call released HOLD: call on hold UNK: unknown
- Start_Time: List[str]: String indicating the time when the call setup was initiated
- Setup_Time: List[float]: Duration of the call setup procedure Unit: s
- Duration: List[float]: Duration of the call Unit: s
- User_From: List[str]: String with the user ID or phone number of the calling party
- User_To: List[str]: String with the user ID or phone number of the called party

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPANalysis:VOIMs:ALL
value: FetchStruct = driver.data.measurement.ipAnalysis.voIms.all.fetch()
```

Queries the call table in the upper part of the 'Voice Over IMS' view. The results are returned row by row (call by call) : <Reliability>, {<ConID>, ..., <UserTo>}call 1, {...}call 2, ..., {...}call n

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.2 Adelay

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:ADELay
STOP:DATA:MEASurement<MeasInstance>:ADELay
ABORt:DATA:MEASurement<MeasInstance>:ADELay
```

class Adelay

Adelay commands group definition. 25 total commands, 7 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:ADELay
driver.data.measurement.adelay.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:ADELay
driver.data.measurement.adelay.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:ADELay
driver.data.measurement.adelay.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:ADELay
driver.data.measurement.adelay.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:ADELay
driver.data.measurement.adelay.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:ADELay
driver.data.measurement.adelay.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.clone()
```

Subgroups

7.4.2.2.1 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:STATe
value: enums.ResourceState = driver.data.measurement.adelay.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.state.clone()
```

Subgroups

7.4.2.2.1.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available
- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:STATe:ALL
value: FetchStruct = driver.data.measurement.adelay.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.2.2 Ulink

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:ULINk
READ:DATA:MEASurement<MeasInstance>:ADELay:ULINk
```

class Ulink

Ulink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:ULINk
value: List[float] = driver.data.measurement.adelay.ulink.fetch()
```

Query the statistical audio delay results for 'Uplink', 'Downlink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:ULINk
value: List[float] = driver.data.measurement.adelay.ulink.read()
```

Query the statistical audio delay results for 'Uplink', 'Downlink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

7.4.2.2.3 Dlink

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:DLINk
READ:DATA:MEASurement<MeasInstance>:ADELay:DLINk
```

class Dlink

Dlink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:DLINk
value: List[float] = driver.data.measurement.adelay.dlink.fetch()
```

Query the statistical audio delay results for 'Uplink', 'Downlink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:DLINk
value: List[float] = driver.data.measurement.adelay.dlink.read()
```

Query the statistical audio delay results for 'Uplink', 'Downlink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

7.4.2.2.4 Loopback

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:LOOPback
READ:DATA:MEASurement<MeasInstance>:ADELay:LOOPback
```

class Loopback

Loopback commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:LOOPback
value: List[float] = driver.data.measurement.adelay.loopback.fetch()
```

Query the statistical audio delay results for 'Uplink', 'Downlink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:LOOPback
value: List[float] = driver.data.measurement.adelay.loopback.read()
```

Query the statistical audio delay results for 'Uplink', 'Downlink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

7.4.2.2.5 TauLink

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:TAULink
READ:DATA:MEASurement<MeasInstance>:ADELay:TAULink
```

class TauLink

TauLink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:TAULink
value: List[float] = driver.data.measurement.adelay.tauLink.fetch()
```

Query the statistical time of arrival results for 'Uplink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:TAULink
value: List[float] = driver.data.measurement.adelay.tauLink.read()
```

Query the statistical time of arrival results for 'Uplink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

7.4.2.2.6 TaLoopback

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:TALoopback
READ:DATA:MEASurement<MeasInstance>:ADELay:TALoopback
```

class TaLoopback

TaLoopback commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:TALoopback
value: List[float] = driver.data.measurement.adelay.taLoopback.fetch()
```

Query the statistical time of arrival results for 'Uplink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:TALoopback
value: List[float] = driver.data.measurement.adelay.taLoopback.read()
```

Query the statistical time of arrival results for 'Uplink' and 'Loopback'.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of four results: Current, Average, Minimum, Maximum value Unit: s

7.4.2.2.7 Trace

class Trace

Trace commands group definition. 10 total commands, 5 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.trace.clone()
```

Subgroups

7.4.2.2.7.1 Ulink

class Ulink

Ulink commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.trace.ulink.clone()
```

Subgroups

7.4.2.2.7.2 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:TRACe:ULINk:CURRent
READ:DATA:MEASurement<MeasInstance>:ADELay:TRACe:ULINk:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:TRACe:ULINk[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.ulink.current.fetch()
```

Query the values of the audio delay traces 'Uplink', 'Downlink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of delay values, one result per sample Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:TRACe:ULINk[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.ulink.current.read()
```

Query the values of the audio delay traces 'Uplink', 'Downlink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of delay values, one result per sample Unit: s

7.4.2.2.7.3 Dlink

class Dlink

Dlink commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.trace.dlink.clone()
```

Subgroups

7.4.2.2.7.4 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:TRACe:DLINk:CURRent
READ:DATA:MEASurement<MeasInstance>:ADELay:TRACe:DLINk:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:TRACe:DLINk[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.dlink.current.fetch()
```

Query the values of the audio delay traces 'Uplink', 'Downlink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of delay values, one result per sample Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:TRACe:DLINk[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.dlink.current.read()
```

Query the values of the audio delay traces 'Uplink', 'Downlink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of delay values, one result per sample Unit: s

7.4.2.2.7.5 Loopback

class Loopback

Loopback commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.trace.loopback.clone()
```

Subgroups

7.4.2.2.7.6 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:TRACe:LOOPback:CURRent
READ:DATA:MEASurement<MeasInstance>:ADELay:TRACe:LOOPback:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:TRACe:LOOPback[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.loopback.current.
↪fetch()
```

Query the values of the audio delay traces 'Uplink', 'Downlink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of delay values, one result per sample Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:TRACe:LOOPback[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.loopback.current.
↪read()
```

Query the values of the audio delay traces 'Uplink', 'Downlink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of delay values, one result per sample Unit: s

7.4.2.2.7.7 TauLink

class TauLink

TauLink commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.trace.tauLink.clone()
```

Subgroups

7.4.2.2.7.8 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:TRACe:TAULink:CURRent
READ:DATA:MEASurement<MeasInstance>:ADELay:TRACe:TAULink:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:TRACe:TAULink[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.tauLink.current.
↪fetch()
```

Query the values of the time of arrival traces 'Uplink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of time error values, one result per sample Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:TRACe:TAULink[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.tauLink.current.read()
```

Query the values of the time of arrival traces 'Uplink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of time error values, one result per sample Unit: s

7.4.2.2.7.9 TaLoopback

class TaLoopback

TaLoopback commands group definition. 2 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.adelay.trace.taLoopback.clone()
```

Subgroups

7.4.2.2.7.10 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:ADELay:TRACe:TALoopback:CURRent
READ:DATA:MEASurement<MeasInstance>:ADELay:TRACe:TALoopback:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:ADELay:TRACe:TALoopback[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.taLoopback.current.
↪fetch()
```

Query the values of the time of arrival traces 'Uplink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of time error values, one result per sample Unit: s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:ADELay:TRACe:TALoopback[:CURRent]
value: List[float] = driver.data.measurement.adelay.trace.taLoopback.current.
↪read()
```

Query the values of the time of arrival traces 'Uplink' and 'Loopback'. The trace values are returned from right to left, from sample number 0 to sample number -N. N equals the configured maximum number of samples minus one, see method RsCmwDau.Configure.Data.Measurement.Adelay.msamples.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of time error values, one result per sample Unit: s

7.4.2.3 Throughput

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:THRoughput
STOP:DATA:MEASurement<MeasInstance>:THRoughput
ABORt:DATA:MEASurement<MeasInstance>:THRoughput
```

class Throughput

Throughput commands group definition. 29 total commands, 4 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:THRoughput
driver.data.measurement.throughput.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:THRoughput
driver.data.measurement.throughput.abort_with_opc()
```

(continues on next page)

(continued from previous page)

```
    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:THRoughput
driver.data.measurement.throughput.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:THRoughput
driver.data.measurement.throughput.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:THRoughput
driver.data.measurement.throughput.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:THRoughput
driver.data.measurement.throughput.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.clone()
```

Subgroups

7.4.2.3.1 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:STATe
value: enums.ResourceState = driver.data.measurement.throughput.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.state.clone()
```

Subgroups

7.4.2.3.1.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:STATe:ALL
value: FetchStruct = driver.data.measurement.throughput.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.3.2 Trace

class Trace

Trace commands group definition. 12 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.trace.clone()
```

Subgroups

7.4.2.3.2.1 Overall

class Overall

Overall commands group definition. 8 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.trace.overall.clone()
```

Subgroups

7.4.2.3.2.2 Dlink

class Dlink
Dlink commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.trace.overall.dlink.clone()
```

Subgroups

7.4.2.3.2.3 Extended

SCPI Commands

```
READ:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:DLINk:EXTended
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:DLINk:EXTended
```

class Extended
Extended commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData
Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Counter: List[int]: Counter for identification of the result value The counter starts with 1 and is incremented for each result as long as the measurement is running.
- Result: List[float]: Unit: bit/s

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:TRACe:OVERall:DLINk:EXTended
value: ResultData = driver.data.measurement.throughput.trace.overall.dlink.
↪extended.fetch()
```

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) . There are two values per interval: <Reliability>, {<Counter>, <Result>}interval n, {...}interval n-1, ... The counter is useful if you want to perform repeated queries and combine the returned result traces. To configure the number of intervals, see method RsCmwDau.Configure.Data.Measurement.Throughput. mcount.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:TRACe:OVERall:DLINk:EXTended
value: ResultData = driver.data.measurement.throughput.trace.overall.dlink.
↪extended.read()
```

(continues on next page)

(continued from previous page)

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) . There are two values per interval: <Reliability>, {<Counter>, <Result>}interval n, {...}interval n-1, ... The counter is useful if you want to perform repeated queries and combine the returned result traces. To configure the number of intervals, see method RsCmwDau.Configure.Data.Measurement.Throughput. mcount.

return structure: for return value, see the help for ResultData structure arguments.

7.4.2.3.2.4 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:DLINk:CURRent
READ:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:DLINk:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:THRoughput:TRACe:OVERall:DLINk[:CURRent]
value: List[float] = driver.data.measurement.throughput.trace.overall.dlink.
↪current.fetch()
```

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau.Configure.Data. Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:TRACe:OVERall:DLINk[:CURRent]
value: List[float] = driver.data.measurement.throughput.trace.overall.dlink.
↪current.read()
```

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau.Configure.Data. Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

7.4.2.3.2.5 Ulink

class Ulink
Ulink commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.trace.overall.ulink.clone()
```

Subgroups

7.4.2.3.2.6 Extended

SCPI Commands

```
READ:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:ULINk:EXTended
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:ULINk:EXTended
```

class Extended
Extended commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData
Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Counter: List[int]: Counter for identification of the result value The counter starts with 1 and is incremented for each result as long as the measurement is running.
- Result: List[float]: Unit: bit/s

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:TRACe:OVERall:ULINk:EXTended
value: ResultData = driver.data.measurement.throughput.trace.overall.ulink.
↪extended.fetch()
```

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) . There are two values per interval: <Reliability>, {<Counter>, <Result>}interval n, {...}interval n-1, ... The counter is useful if you want to perform repeated queries and combine the returned result traces. To configure the number of intervals, see method RsCmwDau.Configure.Data.Measurement.Throughput. mcount.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:TRACe:OVERall:ULINk:EXTended
value: ResultData = driver.data.measurement.throughput.trace.overall.ulink.
↪extended.read()
```

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) . There are two values per interval: <Reliability>, {<Counter>, <Result>}interval n, {...}interval n-1, ... The counter is useful if you want to perform repeated queries and combine the returned result traces. To configure the number of intervals, see method RsCmwDau.Configure.Data.Measurement.Throughput. mcount.

return structure: for return value, see the help for ResultData structure arguments.

7.4.2.3.2.7 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:ULINk:CURRent
READ:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:OVERall:ULINk:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>
↪:THRoughput:TRACe:OVERall:ULINk[:CURRent]
value: List[float] = driver.data.measurement.throughput.trace.overall.ulink.
↪current.fetch()
```

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau.Configure.Data. Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:TRACe:OVERall:ULINk[:CURRent]
value: List[float] = driver.data.measurement.throughput.trace.overall.ulink.
↪current.read()
```

Query the values of the overall throughput trace in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau.Configure.Data. Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

7.4.2.3.2.8 Ran

class Ran
Ran commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.trace.ran.clone()
```

Subgroups

7.4.2.3.2.9 Dlink<Dlink>

RepCap Settings

```
# Range: Ix1 .. Ix4
rc = driver.data.measurement.throughput.trace.ran.dlink.repcap_dlink_get()
driver.data.measurement.throughput.trace.ran.dlink.repcap_dlink_set(repcap.Dlink.Ix1)
```

class Dlink
Dlink commands group definition. 2 total commands, 1 Sub-groups, 0 group commands Repeated Capability: Dlink, default value after init: Dlink.Ix1

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.trace.ran.dlink.clone()
```

Subgroups

7.4.2.3.2.10 Current

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:RAN:DLINk<Dlink>:CURRent
READ:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:RAN:DLINk<Dlink>:CURRent
```

class Current
Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch(*dlink*=<*Dlink.Default: -1*>) → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:TRACe:RAN:DLINk<Index>
↪:CURRent
value: List[float] = driver.data.measurement.throughput.trace.ran.dlink.current.
↪fetch(dlink = repcap.Dlink.Default)
```

Query the values of the throughput trace for RAN slot number <Index> in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau. Configure.Data.Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

param dlink optional repeated capability selector. Default value: Ix1 (settable in the interface 'Dlink')

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

read(*dlink*=<*Dlink.Default: -1*>) → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:TRACe:RAN:DLINk<Index>
↪:CURRent
value: List[float] = driver.data.measurement.throughput.trace.ran.dlink.current.
↪read(dlink = repcap.Dlink.Default)
```

Query the values of the throughput trace for RAN slot number <Index> in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau. Configure.Data.Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

param dlink optional repeated capability selector. Default value: Ix1 (settable in the interface 'Dlink')

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

7.4.2.3.2.11 Ulink<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.data.measurement.throughput.trace.ran.ulink.repcap_slot_get()
driver.data.measurement.throughput.trace.ran.ulink.repcap_slot_set(repcap.Slot.Nr1)
```

class Ulink

Ulink commands group definition. 2 total commands, 1 Sub-groups, 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.data.measurement.throughput.trace.ran.ulink.clone()
```

Subgroups

7.4.2.3.2.12 Current

SCPI Commands

```
READ:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:RAN:ULINk<Slot>:CURRent
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:TRACe:RAN:ULINk<Slot>:CURRent
```

class Current

Current commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch(*slot*=<*Slot.Default: -1*>) → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:TRACe:RAN:ULINk<Index>
↪:CURRent
value: List[float] = driver.data.measurement.throughput.trace.ran.ulink.current.
↪fetch(slot = repcap.Slot.Default)
```

Query the values of the throughput trace for RAN slot number <Index> in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau. Configure.Data.Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

param slot optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ulink')

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

read(*slot*=<*Slot.Default: -1*>) → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:TRACe:RAN:ULINk<Index>
↪:CURRent
value: List[float] = driver.data.measurement.throughput.trace.ran.ulink.current.
↪read(slot = repcap.Slot.Default)
```

Query the values of the throughput trace for RAN slot number <Index> in uplink (ULINk) or downlink (DLINk) direction. The trace values are returned from right to left (last to first measurement) , one result per interval, see method RsCmwDau. Configure.Data.Measurement.Throughput.mcount.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

param slot optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ulink')

return results: Comma-separated list of throughput values, one result per interval Unit: bit/s

7.4.2.3.3 Overall

class Overall

Overall commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.overall.clone()
```

Subgroups

7.4.2.3.3.1 Ulink

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:OVERall:ULINk
READ:DATA:MEASurement<MeasInstance>:THRoughput:OVERall:ULINk
```

class Ulink

Ulink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Th_Curr_Up_Link: float: Current throughput Unit: bit/s
- Th_Min_Up_Link: float: Minimum throughput Unit: bit/s
- Th_Max_Up_Link: float: Maximum throughput Unit: bit/s
- Th_Avg_Up_Link: float: Average throughput Unit: bit/s

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:OVERall:ULINk
value: ResultData = driver.data.measurement.throughput.overall.ulink.fetch()
```

Query the statistical results of the throughput measurement in uplink direction.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:OVERall:ULINk
value: ResultData = driver.data.measurement.throughput.overall.ulink.read()
```

Query the statistical results of the throughput measurement in uplink direction.

return structure: for return value, see the help for ResultData structure arguments.

7.4.2.3.3.2 Dlink

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:OVERall:DLINk
READ:DATA:MEASurement<MeasInstance>:THRoughput:OVERall:DLINk
```

class Dlink

Dlink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Th_Curr_Dlink: float: Current throughput Unit: bit/s
- Th_Min_Dlink: float: Minimum throughput Unit: bit/s
- Th_Max_Dlink: float: Maximum throughput Unit: bit/s
- Th_Avg_Dlink: float: Average throughput Unit: bit/s

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:OVERall:DLINk
value: ResultData = driver.data.measurement.throughput.overall.dlink.fetch()
```

Query the statistical results of the throughput measurement in downlink direction.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:OVERall:DLINk
value: ResultData = driver.data.measurement.throughput.overall.dlink.read()
```

Query the statistical results of the throughput measurement in downlink direction.

return structure: for return value, see the help for ResultData structure arguments.

7.4.2.3.4 Ran

class Ran

Ran commands group definition. 8 total commands, 3 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.ran.clone()
```

Subgroups

7.4.2.3.4.1 Total

class Total
Total commands group definition. 4 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.ran.total.clone()
```

Subgroups

7.4.2.3.4.2 Sum

class Sum
Sum commands group definition. 4 total commands, 2 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.ran.total.sum.clone()
```

Subgroups

7.4.2.3.4.3 Dlink

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:RAN:TOTal:SUM:DLINk
READ:DATA:MEASurement<MeasInstance>:THRoughput:RAN:TOTal:SUM:DLINk
```

class Dlink
Dlink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData
Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Sum_Th_Curr_Dlink: float: Current throughput Unit: bit/s
- Sum_Th_Min_Dlink: float: Minimum throughput Unit: bit/s

- Sum_Th_Max_Dlink: float: Maximum throughput Unit: bit/s
- Sum_Th_Avg_Dlink: float: Average throughput Unit: bit/s

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:RAN:TOTal:SUM:DLINk
value: ResultData = driver.data.measurement.throughput.ran.total.sum.dlink.
↪fetch()
```

Query the statistical results of the throughput measurement for the sum of all RAN slots in downlink direction.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:RAN:TOTal:SUM:DLINk
value: ResultData = driver.data.measurement.throughput.ran.total.sum.dlink.
↪read()
```

Query the statistical results of the throughput measurement for the sum of all RAN slots in downlink direction.

return structure: for return value, see the help for ResultData structure arguments.

7.4.2.3.4.4 Ulink

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:RAN:TOTal:SUM:ULINk
READ:DATA:MEASurement<MeasInstance>:THRoughput:RAN:TOTal:SUM:ULINk
```

class Ulink

Ulink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

class ResultData

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Sum_Th_Curr_Uplink: float: Current throughput Unit: bit/s
- Sum_Th_Min_Uplink: float: Minimum throughput Unit: bit/s
- Sum_Th_Max_Uplink: float: Maximum throughput Unit: bit/s
- Sum_Th_Avg_Uplink: float: Average throughput Unit: bit/s

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:RAN:TOTal:SUM:ULINk
value: ResultData = driver.data.measurement.throughput.ran.total.sum.ulink.
↪fetch()
```

Query the statistical results of the throughput measurement for the sum of all RAN slots in uplink direction.

return structure: for return value, see the help for ResultData structure arguments.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:RAN:TOTal:SUM:ULINk
value: ResultData = driver.data.measurement.throughput.ran.total.sum.ulink.
↪read()
```

Query the statistical results of the throughput measurement for the sum of all RAN slots in uplink direction.

return structure: for return value, see the help for ResultData structure arguments.

7.4.2.3.4.5 Dlink<Dlink>

RepCap Settings

```
# Range: Ix1 .. Ix4
rc = driver.data.measurement.throughput.ran.dlink.repcap_dlink_get()
driver.data.measurement.throughput.ran.dlink.repcap_dlink_set(repcap.Dlink.Ix1)
```

SCPI Commands

```
READ:DATA:MEASurement<MeasInstance>:THRoughput:RAN:DLINk<Dlink>
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:RAN:DLINk<Dlink>
```

class Dlink

Dlink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands Repeated Capability: Dlink, default value after init: Dlink.Ix1

class ResultData

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Th_Curr_Dlink: float: Current throughput Unit: bit/s
- Th_Min_Dlink: float: Minimum throughput Unit: bit/s
- Th_Max_Dlink: float: Maximum throughput Unit: bit/s
- Th_Avg_Dlink: float: Average throughput Unit: bit/s

fetch(*dlink*=<*Dlink.Default: -1*>) → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:RAN:DLINk<Index>
value: ResultData = driver.data.measurement.throughput.ran.dlink.fetch(dlink =␣
↪repcap.Dlink.Default)
```

Query the statistical results of the throughput measurement for RAN slot number <Index> in downlink direction.

param dlink optional repeated capability selector. Default value: Ix1 (settable in the interface 'Dlink')

return structure: for return value, see the help for ResultData structure arguments.

read(*dlink*=<*Dlink.Default: -1*>) → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:RAN:DLINk<Index>
value: ResultData = driver.data.measurement.throughput.ran.dlink.read(dlink =␣
↪repcap.Dlink.Default)
```

Query the statistical results of the throughput measurement for RAN slot number <Index> in downlink direction.

param dlink optional repeated capability selector. Default value: Ix1 (settable in the interface 'Dlink')

return structure: for return value, see the help for ResultData structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.ran.dlink.clone()
```

7.4.2.3.4.6 Ulink<Slot>

RepCap Settings

```
# Range: Nr1 .. Nr4
rc = driver.data.measurement.throughput.ran.ulink.repcap_slot_get()
driver.data.measurement.throughput.ran.ulink.repcap_slot_set(repcap.Slot.Nr1)
```

SCPI Commands

```
READ:DATA:MEASurement<MeasInstance>:THRoughput:RAN:ULINk<Slot>
FETCh:DATA:MEASurement<MeasInstance>:THRoughput:RAN:ULINk<Slot>
```

class Ulink

Ulink commands group definition. 2 total commands, 0 Sub-groups, 2 group commands Repeated Capability: Slot, default value after init: Slot.Nr1

class ResultData

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Th_Curr_Up_Link: float: Current throughput Unit: bit/s
- Th_Min_Up_Link: float: Minimum throughput Unit: bit/s
- Th_Max_Up_Link: float: Maximum throughput Unit: bit/s
- Th_Avg_Up_Link: float: Average throughput Unit: bit/s

fetch(*slot*=<*Slot.Default: -1*>) → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:THRoughput:RAN:ULINk<Index>
value: ResultData = driver.data.measurement.throughput.ran.ulink.fetch(slot =␣
↪repcap.Slot.Default)
```

Query the statistical results of the throughput measurement for RAN slot number <Index> in uplink direction.

param slot optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ulink')

return structure: for return value, see the help for ResultData structure arguments.

read(*slot*=<*Slot.Default: -1*>) → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:THRoughput:RAN:ULINk<Index>
value: ResultData = driver.data.measurement.throughput.ran.ulink.read(slot =␣
↪repcap.Slot.Default)
```

Query the statistical results of the throughput measurement for RAN slot number <Index> in uplink direction.

param slot optional repeated capability selector. Default value: Nr1 (settable in the interface 'Ulink')

return structure: for return value, see the help for ResultData structure arguments.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.throughput.ran.ulink.clone()
```

7.4.2.4 Ping

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:PING
STOP:DATA:MEASurement<MeasInstance>:PING
ABORt:DATA:MEASurement<MeasInstance>:PING
READ:DATA:MEASurement<MeasInstance>:PING
FETCh:DATA:MEASurement<MeasInstance>:PING
```

class Ping

Ping commands group definition. 9 total commands, 3 Sub-groups, 5 group commands

class ResultData

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Req_No: List[int]: Request label, 0 = last request, -1 = previous request, and so on Range: -1000 to 0
- Timestamp: List[str]: Timestamp as string in the format 'hh:mm:ss'
- Latency: List[float]: Round-trip time for sent packets (0 s = no reply) Range: 0 s to 10 s, Unit: s

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:PING
driver.data.measurement.ping.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:PING
driver.data.measurement.ping.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:PING
value: ResultData = driver.data.measurement.ping.fetch()
```

Queries the measured ping characteristics. After the reliability indicator, three results are returned for each ping request: <Reliability>, {<ReqNo>, <Timestamp>, <Latency>}request 1, {...}request 2, ... The number of ping requests is specified by method RsCmwDau.Configure.Data.Measurement.Ping.pcount.

return structure: for return value, see the help for ResultData structure arguments.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:PING
driver.data.measurement.ping.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:PING
driver.data.measurement.ping.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:PING
value: ResultData = driver.data.measurement.ping.read()
```

Queries the measured ping characteristics. After the reliability indicator, three results are returned for each ping request: <Reliability>, {<ReqNo>, <Timestamp>, <Latency>}request 1, {...}request 2, ... The number of ping requests is specified by method RsCmwDau.Configure.Data.Measurement.Ping.pcount.

return structure: for return value, see the help for ResultData structure arguments.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:PING
driver.data.measurement.ping.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:PING
driver.data.measurement.ping.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ping.clone()
```

Subgroups

7.4.2.4.1 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:PING:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:PING:STATe
value: enums.ResourceState = driver.data.measurement.ping.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ping.state.clone()
```

Subgroups

7.4.2.4.1.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:PING:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:PING:STATe:ALL
value: FetchStruct = driver.data.measurement.ping.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.4.2 Overall

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:PING:OVERall
```

class Overall

Overall commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Min_Ping: float: Minimum ping latency Range: 0 s to 10 s, Unit: s
- Max_Ping: float: Maximum ping latency Range: 0 s to 10 s, Unit: s
- Average_Ping: float: Average ping latency Range: 0 s to 10 s, Unit: s

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:PING:OVERall
value: FetchStruct = driver.data.measurement.ping.overall.fetch()
```

Query the statistical ping results over all ping requests.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.4.3 NrCount

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:PING:NRCount
```

class NrCount

NrCount commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → int

```
# SCPI: FETCh:DATA:MEASurement<Instance>:PING:NRCount
value: int = driver.data.measurement.ping.nrCount.fetch()
```

Queries the number of ping requests that have not been answered (no reply) .

return req_no: Range: 0 to 1000

7.4.2.5 DnsRequests

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:DNSRequests
STOP:DATA:MEASurement<MeasInstance>:DNSRequests
ABORt:DATA:MEASurement<MeasInstance>:DNSRequests
```

class DnsRequests

DnsRequests commands group definition. 5 total commands, 1 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:DNSRequests
driver.data.measurement.dnsRequests.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:DNSRequests
driver.data.measurement.dnsRequests.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:DNSRequests
driver.data.measurement.dnsRequests.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:DNSRequests
driver.data.measurement.dnsRequests.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:DNSRequests
driver.data.measurement.dnsRequests.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:DNSRequests
driver.data.measurement.dnsRequests.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.dnsRequests.clone()
```

Subgroups

7.4.2.5.1 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:DNSRequests:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:DNSRequests:STATe
value: enums.ResourceState = driver.data.measurement.dnsRequests.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.dnsRequests.state.clone()
```

Subgroups

7.4.2.5.1.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:DNSRequests:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:DNSRequests:STATe:ALL
value: FetchStruct = driver.data.measurement.dnsRequests.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.6 Iperf

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:IPERf
STOP:DATA:MEASurement<MeasInstance>:IPERf
ABORt:DATA:MEASurement<MeasInstance>:IPERf
READ:DATA:MEASurement<MeasInstance>:IPERf
FETCh:DATA:MEASurement<MeasInstance>:IPERf
```

class Iperf

Iperf commands group definition. 13 total commands, 5 Sub-groups, 5 group commands

class ResultData

Response structure. Fields:

- Reliability: int: No parameter help available
- Up_Bandwidth: float: No parameter help available
- Down_Bandwidth: float: No parameter help available

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPERf
driver.data.measurement.iperf.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPERf
driver.data.measurement.iperf.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

fetch() → ResultData

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPERf
value: ResultData = driver.data.measurement.iperf.fetch()
```

No command help available

return structure: for return value, see the help for ResultData structure arguments.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPERf
driver.data.measurement.iperf.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPERf
driver.data.measurement.iperf.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

read() → ResultData

```
# SCPI: READ:DATA:MEASurement<Instance>:IPERf
value: ResultData = driver.data.measurement.iperf.read()
```

No command help available

return structure: for return value, see the help for ResultData structure arguments.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPERf
driver.data.measurement.iperf.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPERf
driver.data.measurement.iperf.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.iperf.clone()
```

Subgroups

7.4.2.6.1 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPERf:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPERf:STATe
value: enums.ResourceState = driver.data.measurement.iperf.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.iperf.state.clone()
```

Subgroups

7.4.2.6.1.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPERf:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available
- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPERf:STATe:ALL
value: FetchStruct = driver.data.measurement.iperf.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.6.2 PacketLoss

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPERf:PACKetloss
```

class PacketLoss

PacketLoss commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[int]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPERf:PACKetloss
value: List[int] = driver.data.measurement.iperf.packetLoss.fetch()
```

Queries the packet loss for all uplink instances.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return packet_loss: Comma-separated list of eight results (instance 1 to 8) Range: 0 % to 100 %, Unit: %

7.4.2.6.3 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPERf:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Reliability: int: See 'Reliability Indicator'
- Server_Result_Counter: List[int]: No parameter help available
- Client_Result_Counter: List[int]: No parameter help available
- Up_Bandwidth: List[float]: No parameter help available
- Pack_Err_Rate: List[float]: Percentage of lost packets Range: 0 % to 100 %, Unit: %
- Down_Bandwidth: List[float]: No parameter help available

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPERf:ALL
value: FetchStruct = driver.data.measurement.iperf.all.fetch()
```

Queries all client and server results of the iperf measurement. For each server/client instance five results are returned, from instance 1 to instance 8: <Reliability>, {<ServerCounter>, <ClientCounter>, <ServerBW>, <PackErrRate>, <ClientBW>}instance 1, {...}instance 2, ..., {...}instance 8 Iperf results are often queried within a loop, to monitor the results over some time. Iperf delivers new results once per second. If your loop is faster, several consecutive queries deliver the same results. Use the <ServerCounter> and <ClientCounter> to identify redundant results and discard them.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.6.4 Server

SCPI Commands

```
READ:DATA:MEASurement<MeasInstance>:IPERf:SERVer
FETCh:DATA:MEASurement<MeasInstance>:IPERf:SERVer
```

class Server

Server commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPERf:SERVer
value: List[float] = driver.data.measurement.iperf.server.fetch()
```

Queries the throughput for all server instances.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return up_bandwidth: Comma-separated list of eight results (server instance 1 to 8)
Unit: bit/s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:IPERf:SERVer
value: List[float] = driver.data.measurement.iperf.server.read()
```

Queries the throughput for all server instances.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return up_bandwidth: Comma-separated list of eight results (server instance 1 to 8)
Unit: bit/s

7.4.2.6.5 Client

SCPI Commands

```
READ:DATA:MEASurement<MeasInstance>:IPERf:CLIent
FETCh:DATA:MEASurement<MeasInstance>:IPERf:CLIent
```

class Client

Client commands group definition. 2 total commands, 0 Sub-groups, 2 group commands

fetch() → List[float]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPERf:CLIent
value: List[float] = driver.data.measurement.iperf.client.fetch()
```

Queries the throughput for all client instances.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return down_bandwidth: Comma-separated list of eight results (client instance 1 to 8)
Unit: bit/s

read() → List[float]

```
# SCPI: READ:DATA:MEASurement<Instance>:IPERf:CLIent
value: List[float] = driver.data.measurement.iperf.client.read()
```

Queries the throughput for all client instances.

Use RsCmwDau.reliability.last_value to read the updated reliability indicator.

return down_bandwidth: Comma-separated list of eight results (client instance 1 to 8)
Unit: bit/s

7.4.2.7 IpLogging

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:IPLogging
STOP:DATA:MEASurement<MeasInstance>:IPLogging
ABORt:DATA:MEASurement<MeasInstance>:IPLogging
```

class IpLogging

IpLogging commands group definition. 5 total commands, 1 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPLogging
driver.data.measurement.ipLogging.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPLogging
driver.data.measurement.ipLogging.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:
```

(continues on next page)

(continued from previous page)

```
    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPLogging
driver.data.measurement.ipLogging.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPLogging
driver.data.measurement.ipLogging.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPLogging
driver.data.measurement.ipLogging.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPLogging
driver.data.measurement.ipLogging.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipLogging.clone()
```

Subgroups

7.4.2.7.1 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPLogging:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPLogging:STATe
value: enums.ResourceState = driver.data.measurement.ipLogging.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipLogging.state.clone()
```

Subgroups

7.4.2.7.1.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPLogging:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPLogging:STATe:ALL
value: FetchStruct = driver.data.measurement.ipLogging.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.8 IpReplay

SCPI Commands

```
INITiate:DATA:MEASurement<MeasInstance>:IPReplay
STOP:DATA:MEASurement<MeasInstance>:IPReplay
ABORt:DATA:MEASurement<MeasInstance>:IPReplay
```

class IpReplay

IpReplay commands group definition. 6 total commands, 2 Sub-groups, 3 group commands

abort() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPReplay
driver.data.measurement.ipReplay.abort()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

abort_with_opc() → None

```
# SCPI: ABORt:DATA:MEASurement<Instance>:IPReplay
driver.data.measurement.ipReplay.abort_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as abort, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

initiate() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPReplay
driver.data.measurement.ipReplay.initiate()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

initiate_with_opc() → None

```
# SCPI: INITiate:DATA:MEASurement<Instance>:IPReplay
driver.data.measurement.ipReplay.initiate_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

(continues on next page)

(continued from previous page)

Use FETCh...STATe? to query the current measurement state.

Same as initiate, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

stop() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPReplay
driver.data.measurement.ipReplay.stop()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

stop_with_opc() → None

```
# SCPI: STOP:DATA:MEASurement<Instance>:IPReplay
driver.data.measurement.ipReplay.stop_with_opc()


    INTRO_CMD_HELP: Starts, stops, or aborts the measurement:

    - INITiate... starts or restarts the measurement. The measurement enters␣
↪the 'RUN' state.
    - STOP... halts the measurement immediately. The measurement enters the 'RDY
↪' state. Measurement results are kept. The resources remain allocated to the␣
↪measurement.
    - ABORt... halts the measurement immediately. The measurement enters the
↪'OFF' state. All measurement values are set to NAV. Allocated resources are␣
↪released.
```

Use FETCh...STATe? to query the current measurement state.

Same as stop, but waits for the operation to complete before continuing further. Use the RsCmwDau.utilities.opc_timeout_set() to set the timeout value.

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipReplay.clone()
```

Subgroups

7.4.2.8.1 State

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPReplay:STATe
```

class State

State commands group definition. 2 total commands, 1 Sub-groups, 1 group commands

fetch() → RsCmwDau.enums.ResourceState

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPReplay:STATe
value: enums.ResourceState = driver.data.measurement.ipReplay.state.fetch()
```

Queries the main measurement state. Use FETCh:...:STATe:ALL? to query the measurement state including the substates. Use INITiate..., STOP..., ABORt... to change the measurement state.

return meas_state: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.data.measurement.ipReplay.state.clone()
```

Subgroups

7.4.2.8.1.1 All

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPReplay:STATe:ALL
```

class All

All commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class FetchStruct

Response structure. Fields:

- Main_State: enums.ResourceState: OFF | RUN | RDY OFF: measurement off, no resources allocated, no results RUN: measurement running, synchronization pending or adjusted, resources active or queued RDY: measurement terminated, valid results can be available

- Sync_State: enums.ResourceState: PEND | ADJ | INV PEND: waiting for resource allocation, adjustment, hardware switching ('pending') ADJ: adjustments finished, measurement running ('adjusted') INV: not applicable, MainState OFF or RDY ('invalid')
- Resource_State: enums.ResourceState: QUE | ACT | INV QUE: measurement without resources, no results available ('queued') ACT: resources allocated, acquisition of results in progress but not complete ('active') INV: not applicable, MainState OFF or RDY ('invalid')

fetch() → FetchStruct

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPReplay:STATe:ALL
value: FetchStruct = driver.data.measurement.ipReplay.state.all.fetch()
```

Queries the main measurement state and the measurement substates. Both measurement substates are relevant for running measurements only. Use FETCh:...:STATe? to query the main measurement state only. Use INITiate..., STOP..., ABORt... to change the measurement state.

return structure: for return value, see the help for FetchStruct structure arguments.

7.4.2.8.2 FileList

SCPI Commands

```
FETCh:DATA:MEASurement<MeasInstance>:IPReplay:FILelist
```

class FileList

FileList commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

fetch() → List[str]

```
# SCPI: FETCh:DATA:MEASurement<Instance>:IPReplay:FILelist
value: List[str] = driver.data.measurement.ipReplay.fileList.fetch()
```

Queries a list of all files in the directory ip_replay of the samba share.

return files: Comma-separated list of strings, one string per filename

7.5 Rdau

class Rdau

Rdau commands group definition. 1 total commands, 1 Sub-groups, 0 group commands

Cloning the Group

```
# Create a clone of the original group, that exists independently
group2 = driver.rdau.clone()
```

Subgroups

7.5.1 State

SCPI Commands

```
RDAU:STATe
```

class State

State commands group definition. 1 total commands, 0 Sub-groups, 1 group commands

class GetStruct

Response structure. Fields:

- Reliability: int: No parameter help available
- State: enums.DauState: No parameter help available
- Serial_Number: int: No parameter help available
- Ip_Address: str: No parameter help available
- Ref_Count: int: No parameter help available

get() → GetStruct

```
# SCPI: RDAU:STATe
value: GetStruct = driver.rdau.state.get()
```

No command help available

return structure: for return value, see the help for GetStruct structure arguments.

set(*control: bool*, *serial_number: int*) → None

```
# SCPI: RDAU:STATe
driver.rdau.state.set(control = False, serial_number = 1)
```

No command help available

param control No help available

param serial_number No help available

INDEX

F

S