NI-DCPower Python API Documentation

Release 1.4.8

NI

Apr 26, 2024

DOCUMENTATION

CHAPTER

ONE

ABOUT

The **nidcpower** module provides a Python API for NI-DCPower. The code is maintained in the Open Source repository for nimi-python.

1.1 Support Policy

nidcpower supports all the Operating Systems supported by NI-DCPower.

It follows Python Software Foundation support policy for different versions of CPython.

CHAPTER

TWO

CONTRIBUTING

We welcome contributions! You can clone the project repository, build it, and install it by following these instructions.

CHAPTER

THREE

SUPPORT / FEEDBACK

For support specific to the Python API, follow the processs in *Bugs / Feature Requests*. For support with hardware, the driver runtime or any other questions not specific to the Python API, please visit NI Community Forums.

CHAPTER

FOUR

BUGS / FEATURE REQUESTS

To report a bug or submit a feature request specific to Python API, please use the GitHub issues page.

Fill in the issue template as completely as possible and we will respond as soon as we can.

4.1 nidcpower module

4.1.1 Installation

As a prerequisite to using the **nidcpower** module, you must install the NI-DCPower runtime on your system. Visit ni.com/downloads to download the driver runtime for your devices.

The nimi-python modules (i.e. for **NI-DCPower**) can be installed with pip:

```
$ python -m pip install nidcpower~=1.4.8
```

4.1.2 Usage

The following is a basic example of using the **nidcpower** module to open a session to a Source Meter Unit and measure voltage and current.

```
import nidcpower
# Configure the session.

with nidcpower.Session(resource_name='PXI1Slot2/0') as session:
    session.measure_record_length = 20
    session.measure_record_length_is_finite = True
    session.measure_when = nidcpower.MeasureWhen.AUTOMATICALLY_AFTER_SOURCE_COMPLETE
    session.voltage_level = 5.0

    session.commit()
    print('Effective measurement rate: {} S/s'.format(session.measure_record_delta_time /
↪ 1))

    samples_acquired = 0
    print('Channel           Num  Voltage    Current    In Compliance')
    row_format = '{0:15} {1:3d}    {2:8.6f}   {3:8.6f}   {4}'
    with session.initiate():
        channel_indices = '0-{}'.format(session.channel_count - 1)
```

(continues on next page)

(continued from previous page)

```
        channels = session.get_channel_names(channel_indices)
        for i, channel_name in enumerate(channels):
            samples_acquired = 0
            while samples_acquired < 20:
                measurements = session.channels[channel_name].fetch_
↪multiple(count=session.fetch_backlog)
                samples_acquired += len(measurements)
                for i in range(len(measurements)):
                    print(row_format.format(channel_name, i, measurements[i].voltage,␣
↪measurements[i].current, measurements[i].in_compliance))
```

Other usage examples can be found on GitHub.

4.1.3 API Reference

Session

class nidcpower.**Session**(*self*, *resource_name*, *channels=None*, *reset=False*, *options={}*, *independent_channels=True*, ***, *grpc_options=None*)

Creates and returns a new NI-DCPower session to the instrument(s) and channel(s) specified in **resource name** to be used in all subsequent NI-DCPower method calls. With this method, you can optionally set the initial state of the following session properties:

- *nidcpower.Session.simulate*
- *nidcpower.Session.driver_setup*

After calling this method, the specified channel or channels will be in the Uncommitted state.

To place channel(s) in a known start-up state when creating a new session, set **reset** to True. This action is equivalent to using the *nidcpower.Session.reset()* method immediately after initializing the session.

To open a session and leave the channel(s) in an existing configuration without passing through a transitional output state, set **reset** to False. Next, configure the channel(s) as in the previous session, change the desired settings, and then call the *nidcpower.Session.initiate()* method to write both settings.

Details of Independent Channel Operation

With this method and channel-based NI-DCPower methods and properties, you can use any channels in the session independently. For example, you can initiate a subset of channels in the session with *nidcpower.Session.initiate()*, and the other channels in the session remain in the Uncommitted state.

When you initialize with independent channels, each channel steps through the NI-DCPower programming state model independently of all other channels, and you can specify a subset of channels for most operations.

Note You can make concurrent calls to a session from multiple threads, but the session executes the calls one at a time. If you specify multiple channels for a method or property, the session may perform the operation on multiple channels in parallel, though this is not guaranteed, and some operations may execute sequentially.

Parameters

- **resource_name** (*str, list, tuple*) – Specifies the **resource name** as seen in Measurement & Automation Explorer (MAX) or lsni, for example “PXI1Slot3” where “PXI1Slot3” is an instrument’s **resource name**. If independent_channels is False, **resource name** can also be a logical IVI name.

If independent_channels is True, **resource name** can be names of the instrument(s) and the channel(s) to initialize. Specify the instrument(s) and channel(s) using the form "PXI1Slot3/0,PXI1Slot3/2-3,PXI1Slot4/2-3 or PXI1Slot3/0,PXI1Slot3/2:3,PXI1Slot4/2:3", where "PXI1Slot3" and "PXI1Slot4" are instrument resource names followed by channels. If you exclude a channels string after an instrument resource name, all channels of the instrument(s) are included in the session.

- **channels** (*str*, *list*, *range*, *tuple*) – For new applications, use the default value of None and specify the channels in **resource name**.

  Specifies which channel(s) to include in a new session. Specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (for example, 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (for example, 0-2 specifies channels 0, 1, and 2).

  If independent_channels is False, this argument specifies which channels to include in a legacy synchronized channels session. If you do not specify any channels, by default all channels on the device are included in the session.

  If independent_channels is True, this argument combines with **resource name** to specify which channels to include in an independent channels session. Initializing an independent channels session with a channels argument is deprecated.

- **reset** (*bool*) – Specifies whether to reset channel(s) during the initialization procedure.

- **options** (*dict*) – Specifies the initial value of certain properties for the session. The syntax for **options** is a dictionary of properties with an assigned value. For example:

  { 'simulate': False }

  You do not have to specify a value for all the properties. If you do not specify a value for a property, the default value is used.

  Advanced Example: { 'simulate': True, 'driver_setup': { 'Model': '<model number>', 'BoardType': '<type>' } }

Property	Default
range_check	True
query_instrument_status	False
cache	True
simulate	False
record_value_coersions	False
driver_setup	{}

- **independent_channels** (*bool*) – Specifies whether to initialize the session with independent channels. Set this argument to False on legacy applications or if you are unable to upgrade your NI-DCPower driver runtime to 20.6 or higher.

- **grpc_options** (nidcpower.GrpcSessionOptions) – MeasurementLink gRPC session options

Methods

abort

nidcpower.Session.**abort**()

Transitions the specified channel(s) from the Running state to the Uncommitted state. If a sequence is running, it is stopped. Any configuration methods called after this method are not applied until the *nidcpower.Session.initiate()* method is called. If power output is enabled when you call the *nidcpower.Session.abort()* method, the channels remain in their current state and continue providing power.

Use the nidcpower.Session.ConfigureOutputEnabled() method to disable power output on a per channel basis. Use the *nidcpower.Session.reset()* method to disable output on all channels.

Refer to the Programming States topic in the *NI DC Power Supplies and SMUs Help* for information about the specific NI-DCPower software states.

Related Topics:

Programming States

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Note: One or more of the referenced methods are not in the Python API for this driver.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].abort()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.abort()`

---

clear_latched_output_cutoff_state

nidcpower.Session.**clear_latched_output_cutoff_state**(*output_cutoff_reason*)

Clears the state of an output cutoff that was engaged. To clear the state for all output cutoff reasons, use *ALL*.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].clear_latched_output_cutoff_state()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.clear_latched_output_cutoff_state()`

Parameters

output_cutoff_reason (*nidcpower.OutputCutoffReason*) – Specifies the reasons for which to clear the output cutoff state.

ALL	Clears all output cutoff conditions
VOLTAGE_OUTPU	Clears cutoffs caused when the output exceeded the high cutoff limit for voltage output
VOLTAGE_OUTPU	Clears cutoffs caused when the output fell below the low cutoff limit for voltage output
VOLTAGE_MEASU	Clears cutoffs caused when the measured voltage exceeded the high cutoff limit for voltage output
VOLTAGE_MEASU	Clears cutoffs caused when the measured voltage fell below the low cutoff limit for voltage output
CURRENT_MEASU	Clears cutoffs caused when the measured current exceeded the high cutoff limit for current output
CURRENT_MEASU	Clears cutoffs caused when the measured current fell below the low cutoff limit for current output
VOLTAGE_CHANG	Clears cutoffs caused when the voltage slew rate increased beyond the positive change cutoff for voltage output
VOLTAGE_CHANG	Clears cutoffs caused when the voltage slew rate decreased beyond the negative change cutoff for voltage output
CURRENT_CHANG	Clears cutoffs caused when the current slew rate increased beyond the positive change cutoff for current output
CURRENT_CHANG	Clears cutoffs caused when the voltage slew rate decreased beyond the negative change cutoff for current output
CURRENT_SATUR	Clears cutoffs caused when the measured current saturates the current range

close

`nidcpower.Session.close()`

Closes the session specified in **vi** and deallocates the resources that NI-DCPower reserves. If power output is enabled when you call this method, the channels remain in their existing state and continue providing power. Use the `nidcpower.Session.ConfigureOutputEnabled()` method to disable power output on a per channel basis. Use the *nidcpower.Session.reset()* method to disable power output on all channel(s).

Related Topics:

Programming States

Note: NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

Note: One or more of the referenced methods are not in the Python API for this driver.

Note: This method is not needed when using the session context manager

commit

nidcpower.Session.**commit**()

Applies previously configured settings to the specified channel(s). Calling this method moves the NI-DCPower session from the Uncommitted state into the Committed state. After calling this method, modifying any property reverts the NI-DCPower session to the Uncommitted state. Use the *nidcpower.Session.initiate()* method to transition to the Running state. Refer to the Programming States topic in the *NI DC Power Supplies and SMUs Help* for details about the specific NI-DCPower software states.

Related Topics:

Programming States

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].commit()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.commit()`

configure_aperture_time

nidcpower.Session.**configure_aperture_time**(*aperture_time*, *units=nidcpower.ApertureTimeUnits.SECONDS*)

Configures the aperture time on the specified channel(s).

The supported values depend on the **units**. Refer to the *Aperture Time* topic for your device in the *NI DC Power Supplies and SMUs Help* for more information. In general, devices support discrete **apertureTime** values, and if you configure **apertureTime** to some unsupported value, NI-DCPower coerces it up to the next supported value.

Refer to the *Measurement Configuration and Timing* or *DC Noise Rejection* topic for your device in the *NI DC Power Supplies and SMUs Help* for more information about how to configure your measurements.

Related Topics:

Aperture Time

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].configure_aperture_time()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.configure_aperture_time()`

Parameters

- **aperture_time** (*float*) – Specifies the aperture time. Refer to the *Aperture Time* topic for your device in the *NI DC Power Supplies and SMUs Help* for more information.
- **units** (*nidcpower.ApertureTimeUnits*) – Specifies the units for **apertureTime**. **Defined Values**:

SECONDS	Specifies seconds.
POWER_LINE_CYCLES	Specifies Power Line Cycles.

configure_lcr_compensation

nidcpower.Session.**configure_lcr_compensation**(*compensation_data*)

Applies previously generated open, short, load, as well as open and short custom cable compensation data to LCR measurements.

This method applies open, short and load compensation data when you have set the *nidcpower.Session.lcr_open_short_load_compensation_data_source* property to *AS_CONFIGURED*, and it also applies custom cable compensation data when you have set the *nidcpower.Session.cable_length* property to *CUSTOM_AS_CONFIGURED*.

Call this method after you have obtained LCR compensation data.

If the *nidcpower.Session.lcr_short_custom_cable_compensation_enabled* property is set to True, you must generate data with both *nidcpower.Session.perform_lcr_open_custom_cable_compensation()* and *nidcpower.Session.perform_lcr_short_custom_cable_compensation()*; if False, you must only use *nidcpower.Session.perform_lcr_open_custom_cable_compensation()*, and NI-DCPower uses default short data.

Call *nidcpower.Session.get_lcr_compensation_data()* and pass the **compensation data** to this method.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].configure_lcr_compensation()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.configure_lcr_compensation()`

Parameters

compensation_data (*bytes*) – The open, short and load compensation data to apply.

configure_lcr_custom_cable_compensation

nidcpower.Session.**configure_lcr_custom_cable_compensation**(*custom_cable_compensation_data*)

This method is deprecated. Use *nidcpower.Session.configure_lcr_compensation()* instead.

Applies previously generated open and short custom cable compensation data to LCR measurements.

This method applies custom cable compensation data when you have set *nidcpower.Session.cable_length* property to *CUSTOM_AS_CONFIGURED*.

Call this method after you have obtained custom cable compensation data.

If *nidcpower.Session.lcr_short_custom_cable_compensation_enabled* property is set to True, you must generate data with both *nidcpower.Session.perform_lcr_open_custom_cable_compensation()* and *nidcpower.Session.perform_lcr_short_custom_cable_compensation()*; if False, you must only use *nidcpower.Session.perform_lcr_open_custom_cable_compensation()*, and NI-DCPower uses default short data.

Call *nidcpower.Session.get_lcr_custom_cable_compensation_data()* and pass the **custom cable compensation data** to this method.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].configure_lcr_custom_cable_compensation()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.configure_lcr_custom_cable_compensation()`

Parameters

custom_cable_compensation_data (*bytes*) – The open and short custom cable compensation data to apply.

create_advanced_sequence

nidcpower.Session.**create_advanced_sequence**(*sequence_name*, *property_names*, *set_as_active_sequence=True*)

Creates an empty advanced sequence. Call the *nidcpower.Session.create_advanced_sequence_step()* method to add steps to the active advanced sequence.

You can create multiple advanced sequences in a session.

Support for this method

You must set the source mode to Sequence to use this method.

Using the *nidcpower.Session.set_sequence()* method with Advanced Sequence methods is unsupported.

Use this method in the Uncommitted or Committed programming states. Refer to the Programming States topic in the *NI DC Power Supplies and SMUs Help* for more information about NI-DCPower programming states.

Related Topics:

Advanced Sequence Mode

Programming States

nidcpower.Session.create_advanced_sequence_step()

Note: This method is not supported on all devices. Refer to Supported Methods by Device for more information about supported devices.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].create_advanced_sequence()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.create_advanced_sequence()`

Parameters

- **sequence_name** (*str*) – Specifies the name of the sequence to create.
- **property_names** (*list of str*) – Specifies the names of the properties you reconfigure per step in the advanced sequence. The following table lists which properties can be configured in an advanced sequence for each NI-DCPower device that supports advanced sequencing. A Yes indicates that the property can be configured in advanced sequencing. An No indicates that the property cannot be configured in advanced sequencing.

Property	PXIe-4135	PXIe-4136	PXIe-4137
nidcpower.Session.aperture_time	Yes	Yes	Yes
nidcpower.Session.dc_noise_rejection	Yes	No	Yes
nidcpower.Session.instrument_mode	No	No	No
nidcpower.Session.lcr_actual_load_reactance	No	No	No
nidcpower.Session.lcr_actual_load_resistance	No	No	No
nidcpower.Session.lcr_current_amplitude	No	No	No
nidcpower.Session.lcr_current_range	No	No	No
nidcpower.Session.lcr_custom_measurement_time	No	No	No
nidcpower.Session.lcr_dc_bias_current_level	No	No	No
nidcpower.Session.lcr_dc_bias_current_range	No	No	No
nidcpower.Session.lcr_dc_bias_source	No	No	No
nidcpower.Session.lcr_dc_bias_voltage_level	No	No	No
nidcpower.Session.lcr_dc_bias_voltage_range	No	No	No

Table 1 –

Property	PXIe-4135	PXIe-4136	PXIe-4137
nidcpower.Session.lcr_frequency	No	No	No
nidcpower.Session.lcr_impedance_auto_range	No	No	No
nidcpower.Session.lcr_impedance_range	No	No	No
nidcpower.Session.lcr_load_compensation_enabled	No	No	No
nidcpower.Session.lcr_measured_load_reactance	No	No	No
nidcpower.Session.lcr_measured_load_resistance	No	No	No
nidcpower.Session.lcr_measurement_time	No	No	No
nidcpower.Session.lcr_open_compensation_enabled	No	No	No
nidcpower.Session.lcr_open_conductance	No	No	No
nidcpower.Session.lcr_open_susceptance	No	No	No
nidcpower.Session.lcr_short_compensation_enabled	No	No	No
nidcpower.Session.lcr_short_reactance	No	No	No
nidcpower.Session.lcr_short_resistance	No	No	No
nidcpower.Session.lcr_source_delay_mode	No	No	No
nidcpower.Session.lcr_stimulus_function	No	No	No
nidcpower.Session.lcr_voltage_amplitude	No	No	No
nidcpower.Session.lcr_voltage_range	No	No	No
nidcpower.Session.measure_record_length	Yes	Yes	Yes
nidcpower.Session.sense	Yes	Yes	Yes
nidcpower.Session.ovp_enabled	Yes	Yes	Yes
nidcpower.Session.ovp_limit	Yes	Yes	Yes
nidcpower.Session.pulse_bias_delay	Yes	Yes	Yes
nidcpower.Session.pulse_off_time	Yes	Yes	Yes
nidcpower.Session.pulse_on_time	Yes	Yes	Yes
nidcpower.Session.source_delay	Yes	Yes	Yes
nidcpower.Session.current_compensation_frequency	Yes	No	Yes
nidcpower.Session.current_gain_bandwidth	Yes	No	Yes
nidcpower.Session.current_pole_zero_ratio	Yes	No	Yes
nidcpower.Session.voltage_compensation_frequency	Yes	No	Yes
nidcpower.Session.voltage_gain_bandwidth	Yes	No	Yes
nidcpower.Session.voltage_pole_zero_ratio	Yes	No	Yes
nidcpower.Session.current_level	Yes	Yes	Yes
nidcpower.Session.current_level_range	Yes	Yes	Yes
nidcpower.Session.voltage_limit	Yes	Yes	Yes
nidcpower.Session.voltage_limit_high	Yes	Yes	Yes
nidcpower.Session.voltage_limit_low	Yes	Yes	Yes
nidcpower.Session.voltage_limit_range	Yes	Yes	Yes
nidcpower.Session.current_limit	Yes	Yes	Yes
nidcpower.Session.current_limit_high	Yes	Yes	Yes
nidcpower.Session.current_limit_low	Yes	Yes	Yes
nidcpower.Session.current_limit_range	Yes	Yes	Yes
nidcpower.Session.voltage_level	Yes	Yes	Yes
nidcpower.Session.voltage_level_range	Yes	Yes	Yes
nidcpower.Session.output_enabled	Yes	Yes	Yes
nidcpower.Session.output_function	Yes	Yes	Yes
nidcpower.Session.output_resistance	Yes	No	Yes
nidcpower.Session.pulse_bias_current_level	Yes	Yes	Yes
nidcpower.Session.pulse_bias_voltage_limit	Yes	Yes	Yes
nidcpower.Session.pulse_bias_voltage_limit_high	Yes	Yes	Yes
nidcpower.Session.pulse_bias_voltage_limit_low	Yes	Yes	Yes

Table 1 –

Property	PXIe-4135	PXIe-4136	PXIe-4137
nidcpower.Session.pulse_current_level	Yes	Yes	Yes
nidcpower.Session.pulse_current_level_range	Yes	Yes	Yes
nidcpower.Session.pulse_voltage_limit	Yes	Yes	Yes
nidcpower.Session.pulse_voltage_limit_high	Yes	Yes	Yes
nidcpower.Session.pulse_voltage_limit_low	Yes	Yes	Yes
nidcpower.Session.pulse_voltage_limit_range	Yes	Yes	Yes
nidcpower.Session.pulse_bias_current_limit	Yes	Yes	Yes
nidcpower.Session.pulse_bias_current_limit_high	Yes	Yes	Yes
nidcpower.Session.pulse_bias_current_limit_low	Yes	Yes	Yes
nidcpower.Session.pulse_bias_voltage_level	Yes	Yes	Yes
nidcpower.Session.pulse_current_limit	Yes	Yes	Yes
nidcpower.Session.pulse_current_limit_high	Yes	Yes	Yes
nidcpower.Session.pulse_current_limit_low	Yes	Yes	Yes
nidcpower.Session.pulse_current_limit_range	Yes	Yes	Yes
nidcpower.Session.pulse_voltage_level	Yes	Yes	Yes
nidcpower.Session.pulse_voltage_level_range	Yes	Yes	Yes
nidcpower.Session.transient_response	Yes	Yes	Yes

- **set_as_active_sequence** (*bool*) – Specifies that this current sequence is active.

create_advanced_sequence_commit_step

nidcpower.Session.**create_advanced_sequence_commit_step**(*set_as_active_step=True*)

Creates a Commit step in the Active advanced sequence. A Commit step configures channels to a user-defined known state before starting the advanced sequence. When a Commit step exists in the Active advanced sequence, you cannot set the output method to Pulse Voltage or Pulse Current in either the Commit step (-1) or step 0. When you create an advanced sequence step, each property you passed to the *nidcpower.Session.create_advanced_sequence()* method is reset to its default value for that step unless otherwise specified.

Support for this Method

You must set the source mode to Sequence to use this method.

Using the *nidcpower.Session.set_sequence()* method with Advanced Sequence methods is unsupported.

Related Topics:

Advanced Sequence Mode

Programming States

nidcpower.Session.create_advanced_sequence()

Note: NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].create_advanced_sequence_commit_step()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.create_advanced_sequence_commit_step()`

Parameters

set_as_active_step (*bool*) – Specifies whether the step created with this method is active in the Active advanced sequence.

create_advanced_sequence_step

nidcpower.Session.**create_advanced_sequence_step**(*set_as_active_step=True*)

Creates a new advanced sequence step in the advanced sequence specified by the Active advanced sequence. When you create an advanced sequence step, each property you passed to the *nidcpower.Session.create_advanced_sequence()* method is reset to its default value for that step unless otherwise specified.

Support for this Method

You must set the source mode to Sequence to use this method.

Using the *nidcpower.Session.set_sequence()* method with Advanced Sequence methods is unsupported.

Related Topics:

Advanced Sequence Mode

Programming States

nidcpower.Session.create_advanced_sequence()

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].create_advanced_sequence_step()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.create_advanced_sequence_step()`

Parameters

set_as_active_step (*bool*) – Specifies whether the step created with this method is active in the Active advanced sequence.

delete_advanced_sequence

nidcpower.Session.**delete_advanced_sequence**(*sequence_name*)

Deletes a previously created advanced sequence and all the advanced sequence steps in the advanced sequence.

Support for this Method

You must set the source mode to Sequence to use this method.

Using the *nidcpower.Session.set_sequence()* method with Advanced Sequence methods is unsupported.

Related Topics:

Advanced Sequence Mode

Programming States

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].delete_advanced_sequence()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.delete_advanced_sequence()`

Parameters

sequence_name (*str*) – specifies the name of the sequence to delete.

disable

nidcpower.Session.**disable**()

This method performs the same actions as the *nidcpower.Session.reset()* method, except that this method also immediately sets the *nidcpower.Session.output_enabled* property to False.

This method opens the output relay on devices that have an output relay.

Note: NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

---

export_attribute_configuration_buffer

nidcpower.Session.**export_attribute_configuration_buffer**()

Exports the property configuration of the session to the specified configuration buffer.

You can export and import session property configurations only between devices with identical model numbers and the same number of configured channels.

This method verifies that the properties you have configured for the session are valid. If the configuration is invalid, NI-DCPower returns an error.

Support for this Method

Calling this method in Sequence Source Mode is unsupported.

Channel Mapping Behavior for Multichannel Sessions

When importing and exporting session property configurations between NI-DCPower sessions that were initialized with different channels, the configurations of the exporting channels are mapped to the importing channels in the order you specify in the **channelName** input to the `nidcpower.Session.__init__()` method.

For example, if your entry for **channelName** is 0,1 for the exporting session and 1,2 for the importing session:

- The configuration exported from channel 0 is imported into channel 1.
- The configuration exported from channel 1 is imported into channel 2.

Related Topics:

Using Properties and Properties

Setting Properties and Properties Before Reading Them

Note: This method will return an error if the total number of channels initialized for the exporting session is not equal to the total number of channels initialized for the importing session.

Return type
bytes

Returns
Specifies the byte array buffer to be populated with the exported property configuration.

export_attribute_configuration_file

nidcpower.Session.**export_attribute_configuration_file**(*file_path*)

Exports the property configuration of the session to the specified file.

You can export and import session property configurations only between devices with identical model numbers and the same number of configured channels.

This method verifies that the properties you have configured for the session are valid. If the configuration is invalid, NI-DCPower returns an error.

Support for this Method

Calling this method in Sequence Source Mode is unsupported.

Channel Mapping Behavior for Multichannel Sessions

When importing and exporting session property configurations between NI-DCPower sessions that were initialized with different channels, the configurations of the exporting channels are mapped to the importing channels in the order you specify in the **channelName** input to the `nidcpower.Session.__init__()` method.

For example, if your entry for **channelName** is 0,1 for the exporting session and 1,2 for the importing session:

- The configuration exported from channel 0 is imported into channel 1.
- The configuration exported from channel 1 is imported into channel 2.

Related Topics:

Using Properties and Properties

Setting Properties and Properties Before Reading Them

Note: This method will return an error if the total number of channels initialized for the exporting session is not equal to the total number of channels initialized for the importing session.

Parameters

file_path (*str*) – Specifies the absolute path to the file to contain the exported property configuration. If you specify an empty or relative path, this method returns an error. **Default file extension:** .nidcpowerconfig

fetch_multiple

nidcpower.Session.**fetch_multiple**(*count*, *timeout=hightime.timedelta(seconds=1.0)*)

Returns a list of named tuples (Measurement) that were previously taken and are stored in the NI-DCPower buffer. This method should not be used when the *nidcpower.Session.measure_when* property is set to *ON_DEMAND*. You must first call *nidcpower.Session.initiate()* before calling this method.

Fields in Measurement:

- **voltage** (float)
- **current** (float)
- **in_compliance** (bool)
- **channel** (str)

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].fetch_multiple()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.fetch_multiple()`

Parameters

- **count** (*int*) – Specifies the number of measurements to fetch.
- **timeout** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Specifies the maximum time allowed for this method to complete. If the method does not complete within this time interval, NI-DCPower returns an error. Default value: 1.0 second

  Note: When setting the timeout interval, ensure you take into account any triggers so that the timeout interval is long enough for your application.

Return type
list of Measurement

Returns

List of named tuples with fields:

- **voltage** (float)
- **current** (float)
- **in_compliance** (bool)
- **channel** (str)

fetch_multiple_lcr

`nidcpower.Session.`**`fetch_multiple_lcr`**(*count*, *timeout=hightime.timedelta(seconds=1.0)*)

Returns a list of previously measured LCRMeasurement instances on the specified channel that have been taken and stored in a buffer.

To use this method:

- Set *nidcpower.Session.measure_when* property to *AUTOMATICALLY_AFTER_SOURCE_COMPLETE* or *ON_MEASURE_TRIGGER*
- Put the channel in the Running state (call *nidcpower.Session.initiate()*)

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].fetch_multiple_lcr()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.fetch_multiple_lcr()`

Parameters

- **count** (*int*) – Specifies the number of measurements to fetch.
- **timeout** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Specifies the maximum time allowed for this method to complete, in seconds. If the method does not complete within this time interval, NI-DCPower returns an error. Default value: 1.0 second

  Note: When setting the timeout interval, ensure you take into account any triggers so that the timeout interval is long enough for your application.

Return type
list of LCRMeasurement

Returns

A list of LCRMeasurement instances.

channel		The channel name associated with this LCR measurement.
vdc	float	The measured DC voltage, in volts.
idc	float	The measured DC current, in amps.
stimulus_frequ	float	The frequency of the LCR test signal, in Hz.
ac_voltag	complex	The measured AC voltage, in volts RMS.
ac_currer	complex	The measured AC current, in amps RMS.
z	complex	The complex impedance.
z_magnit	tuple of float	The magnitude, in ohms, and phase angle, in degrees, of the complex impedance.
y	complex	The complex admittance.
y_magnit	tuple of float	The magnitude, in siemens, and phase angle, in degrees, of the complex admittance.
series_lcr	LCR	The inductance, in henrys, the capacitance, in farads, and the resistance, in ohms, as measured using a series circuit model.
parallel_lcr	LCR	The inductance, in henrys, the capacitance, in farads, and the resistance, in ohms, as measured using a parallel circuit model.
d	float	The dissipation factor of the circuit. The dimensionless dissipation factor is directly proportional to how quickly an oscillating system loses energy. D is the reciprocal of Q, the quality factor.
q	float	The quality factor of the circuit. The dimensionless quality factor is inversely proportional to the degree of damping in a system. Q is the reciprocal of D, the dissipation factor.
measurement_mo	enums.In	The measurement mode: **SMU** - The channel(s) are operating as a power supply/SMU. **LCR** - The channel(s) are operating as an LCR meter.
dc_in_co	bool	Indicates whether the output was in DC compliance at the time the measurement was taken.
ac_in_co	bool	Indicates whether the output was in AC compliance at the time the measurement was taken.
unbalanced	bool	Indicates whether the output was unbalanced at the time the measurement was taken.

get_channel_name

nidcpower.Session.**get_channel_name**(*index*)

Retrieves the output **channelName** that corresponds to the requested **index**. Use the *nidcpower.Session.channel_count* property to determine the upper bound of valid values for **index**.

Parameters

index (*int*) – Specifies which channel name to return. The index values begin at 1.

Return type

str

Returns

Returns the channel name that corresponds to **index**.

get_channel_names

nidcpower.Session.**get_channel_names**(*indices*)

Returns a list of channel names for the given channel indices.

Parameters

indices (*basic sequence types or str or int*) – Index list for the channels in the session. Valid values are from zero to the total number of channels in the session minus one. The index string can be one of the following formats:

- A comma-separated list—for example, "0,2,3,1"
- A range using a hyphen—for example, "0-3"
- A range using a colon—for example, "0:3 "

You can combine comma-separated lists and ranges that use a hyphen or colon. Both out-of-order and repeated indices are supported ("2,3,0," "1,2,2,3"). White space characters, including spaces, tabs, feeds, and carriage returns, are allowed between characters. Ranges can be incrementing or decrementing.

Return type

list of str

Returns

The channel name(s) at the specified indices.

get_ext_cal_last_date_and_time

nidcpower.Session.**get_ext_cal_last_date_and_time**()

Returns the date and time of the last successful calibration.

Return type

hightime.datetime

Returns

Indicates date and time of the last calibration.

get_ext_cal_last_temp

nidcpower.Session.**get_ext_cal_last_temp**()

Returns the onboard **temperature** of the device, in degrees Celsius, during the last successful external calibration.

Return type

float

Returns

Returns the onboard **temperature** of the device, in degrees Celsius, during the last successful external calibration.

get_ext_cal_recommended_interval

nidcpower.Session.**get_ext_cal_recommended_interval**()

Returns the recommended maximum interval, in **months**, between external calibrations.

Return type
hightime.timedelta

Returns
Specifies the recommended maximum interval, in **months**, between external calibrations.

get_lcr_compensation_data

nidcpower.Session.**get_lcr_compensation_data**()

Collects previously generated open, short, load, and custom cable compensation data so you can then apply it to LCR measurements with *nidcpower.Session.configure_lcr_compensation()*.

Call this method after you have obtained the compensation data of all types (open, short, load, open custom cable compensation, and short custom cable compensation) you want to apply to your measurements. Pass the **compensation data** to *nidcpower.Session.configure_lcr_compensation()*

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].get_lcr_compensation_data()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.get_lcr_compensation_data()`

Return type
bytes

Returns
The open, short, load, and custom cable compensation data to retrieve.

get_lcr_compensation_last_date_and_time

nidcpower.Session.**get_lcr_compensation_last_date_and_time**(*compensation_type*)

Returns the date and time the specified type of compensation data for LCR measurements was most recently generated.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].get_lcr_compensation_last_date_and_time()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.get_lcr_compensation_last_date_and_time()`

Parameters

compensation_type (*nidcpower.LCRCompensationType*) – Specifies the type of compensation for LCR measurements.

Return type

hightime.datetime

Returns

Returns the date and time the specified type of compensation data for LCR measurements was most recently generated.

get_lcr_custom_cable_compensation_data

nidcpower.Session.**get_lcr_custom_cable_compensation_data**()

This method is deprecated. Use *nidcpower.Session.get_lcr_compensation_data()* instead.

Collects previously generated open and short custom cable compensation data so you can then apply it to LCR measurements with *nidcpower.Session.configure_lcr_custom_cable_compensation()*.

Call this method after you have obtained open and short custom cable compensation data. Pass the **custom cable compensation data** to *nidcpower.Session.configure_lcr_custom_cable_compensation()*

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].get_lcr_custom_cable_compensation_data()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.get_lcr_custom_cable_compensation_data()`

Return type

bytes

Returns

The open and short custom cable compensation data to retrieve.

get_self_cal_last_date_and_time

nidcpower.Session.**get_self_cal_last_date_and_time**()

Returns the date and time of the oldest successful self-calibration from among the channels in the session.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Return type
hightime.datetime

Returns
Returns the date and time the device was last calibrated.

get_self_cal_last_temp

nidcpower.Session.**get_self_cal_last_temp**()

Returns the onboard temperature of the device, in degrees Celsius, during the oldest successful self-calibration from among the channels in the session.

For example, if you have a session using channels 1 and 2, and you perform a self-calibration on channel 1 with a device temperature of 25 degrees Celsius at 2:00, and a self-calibration was performed on channel 2 at 27 degrees Celsius at 3:00 on the same day, this method returns 25 for the **temperature** parameter.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Return type
float

Returns
Returns the onboard **temperature** of the device, in degrees Celsius, during the oldest successful calibration.

import_attribute_configuration_buffer

nidcpower.Session.**import_attribute_configuration_buffer**(*configuration*)

Imports a property configuration to the session from the specified configuration buffer.

You can export and import session property configurations only between devices with identical model numbers and the same number of configured channels.

Support for this Method

Calling this method in Sequence Source Mode is unsupported.

Channel Mapping Behavior for Multichannel Sessions

When importing and exporting session property configurations between NI-DCPower sessions that were initialized with different channels, the configurations of the exporting channels are mapped

to the importing channels in the order you specify in the **channelName** input to the `nidcpower.Session.__init__()` method.

For example, if your entry for **channelName** is 0,1 for the exporting session and 1,2 for the importing session:

- The configuration exported from channel 0 is imported into channel 1.
- The configuration exported from channel 1 is imported into channel 2.

Related Topics:

Programming States

Using Properties and Properties

Setting Properties and Properties Before Reading Them

Note: This method will return an error if the total number of channels initialized for the exporting session is not equal to the total number of channels initialized for the importing session.

Parameters

configuration (*bytes*) – Specifies the byte array buffer that contains the property configuration to import.

import_attribute_configuration_file

`nidcpower.Session.`**`import_attribute_configuration_file`**(*file_path*)

Imports a property configuration to the session from the specified file.

You can export and import session property configurations only between devices with identical model numbers and the same number of configured channels.

Support for this Method

Calling this method in Sequence Source Mode is unsupported.

Channel Mapping Behavior for Multichannel Sessions

When importing and exporting session property configurations between NI-DCPower sessions that were initialized with different channels, the configurations of the exporting channels are mapped to the importing channels in the order you specify in the **channelName** input to the `nidcpower.Session.__init__()` method.

For example, if your entry for **channelName** is 0,1 for the exporting session and 1,2 for the importing session:

- The configuration exported from channel 0 is imported into channel 1.
- The configuration exported from channel 1 is imported into channel 2.

Related Topics:

Programming States

Using Properties and Properties

Setting Properties and Properties Before Reading Them

Note: This method will return an error if the total number of channels initialized for the exporting session is not equal to the total number of channels initialized for the importing session.

Parameters

file_path (*str*) – Specifies the absolute path to the file containing the property configuration to import. If you specify an empty or relative path, this method returns an error. **Default File Extension:** .nidcpowerconfig

initiate

nidcpower.Session.**initiate**()

Starts generation or acquisition, causing the specified channel(s) to leave the Uncommitted state or Committed state and enter the Running state. To return to the Uncommitted state call the *nidcpower.Session.abort()* method. Refer to the Programming States topic in the *NI DC Power Supplies and SMUs Help* for information about the specific NI-DCPower software states.

Related Topics:

Programming States

Note: This method will return a Python context manager that will initiate on entering and abort on exit.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].initiate()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.initiate()`

lock

nidcpower.Session.**lock**()

Obtains a multithread lock on the device session. Before doing so, the software waits until all other execution threads release their locks on the device session.

Other threads may have obtained a lock on this session for the following reasons:

- The application called the *nidcpower.Session.lock()* method.
- A call to NI-DCPower locked the session.
- After a call to the *nidcpower.Session.lock()* method returns successfully, no other threads can access the device session until you call the *nidcpower.Session.unlock()* method or exit out of the with block when using lock context manager.
- Use the *nidcpower.Session.lock()* method and the *nidcpower.Session.unlock()* method around a sequence of calls to instrument driver methods if you require that the device retain its settings through the end of the sequence.

You can safely make nested calls to the *nidcpower.Session.lock()* method within the same thread. To completely unlock the session, you must balance each call to the *nidcpower.Session.lock()* method with a call to the *nidcpower.Session.unlock()* method.

One method for ensuring there are the same number of unlock method calls as there is lock calls is to use lock as a context manager

```
with nidcpower.Session('dev1') as session:
    with session.lock():
        # Calls to session within a single lock context
```

The first *with* block ensures the session is closed regardless of any exceptions raised

The second *with* block ensures that unlock is called regardless of any exceptions raised

Return type
context manager

Returns
When used in a *with* statement, *nidcpower.Session.lock()* acts as a context manager and unlock will be called when the *with* block is exited

measure

nidcpower.Session.**measure**(*measurement_type*)

Returns the measured value of either the voltage or current on the specified channel. Each call to this method blocks other method calls until the hardware returns the **measurement**. To measure multiple channels, use the *nidcpower.Session.measure_multiple()* method.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].measure()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.measure()`

Parameters
measurement_type (*nidcpower.MeasurementTypes*) – Specifies whether a voltage or current value is measured. **Defined Values**:

VOLTAGE	The device measures voltage.
CURRENT	The device measures current.

Return type
float

Returns
Returns the value of the measurement, either in volts for voltage or amps for current.

measure_multiple

nidcpower.Session.**measure_multiple**()

Returns a list of named tuples (Measurement) containing the measured voltage and current values on the specified channel(s). Each call to this method blocks other method calls until the measurements are returned from the device. The order of the measurements returned in the array corresponds to the order on the specified channel(s).

Fields in Measurement:

- **voltage** (float)
- **current** (float)
- **in_compliance** (bool) - Always None
- **channel** (str)

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].measure_multiple()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.measure_multiple()`

Return type
list of Measurement

Returns

List of named tuples with fields:

- **voltage** (float)
- **current** (float)
- **in_compliance** (bool) - Always None
- **channel** (str)

measure_multiple_lcr

nidcpower.Session.**measure_multiple_lcr**()

Measures and returns a list of LCRMeasurement instances on the specified channel(s).

To use this method:

- Set *nidcpower.Session.instrument_mode* property to *LCR*
- Set *nidcpower.Session.measure_when* property to *ON_DEMAND*
- Put the channel(s) in the Running state (call *nidcpower.Session.initiate()*)

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].measure_multiple_lcr()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.measure_multiple_lcr()`

Return type
list of LCRMeasurement

Returns

A list of LCRMeasurement instances.

channel		The channel name associated with this LCR measurement.
vdc	float	The measured DC voltage, in volts.
idc	float	The measured DC current, in amps.
stimulus_frequ	float	The frequency of the LCR test signal, in Hz.
ac_voltag	complex	The measured AC voltage, in volts RMS.
ac_currer	complex	The measured AC current, in amps RMS.
z	complex	The complex impedance.
z_magnit	tuple of float	The magnitude, in ohms, and phase angle, in degrees, of the complex impedance.
y	complex	The complex admittance.
y_magnit	tuple of float	The magnitude, in siemens, and phase angle, in degrees, of the complex admittance.
series_lcr	LCR	The inductance, in henrys, the capacitance, in farads, and the resistance, in ohms, as measured using a series circuit model.
parallel_lcr	LCR	The inductance, in henrys, the capacitance, in farads, and the resistance, in ohms, as measured using a parallel circuit model.
d	float	The dissipation factor of the circuit. The dimensionless dissipation factor is directly proportional to how quickly an oscillating system loses energy. D is the reciprocal of Q, the quality factor.
q	float	The quality factor of the circuit. The dimensionless quality factor is inversely proportional to the degree of damping in a system. Q is the reciprocal of D, the dissipation factor.
measurement_mo	enums.In	The measurement mode: **SMU** - The channel(s) are operating as a power supply/SMU. **LCR** - The channel(s) are operating as an LCR meter.
dc_in_co	bool	Indicates whether the output was in DC compliance at the time the measurement was taken.
ac_in_co	bool	Indicates whether the output was in AC compliance at the time the measurement was taken.
unbalanced	bool	Indicates whether the output was unbalanced at the time the measurement was taken.

perform_lcr_load_compensation

nidcpower.Session.**perform_lcr_load_compensation**(*compensation_spots*)

Generates load compensation data for LCR measurements for the test spots you specify.

You must physically configure your LCR circuit with an appropriate reference load to use this method to generate valid load compensation data.

When you call this method:

- The load compensation data is written to the onboard storage of the instrument. Onboard storage can contain only the most recent set of data.
- Most NI-DCPower properties in the session are reset to their default values. Rewrite the values

of any properties you want to maintain.

To apply the load compensation data you generate with this method to your LCR measurements, set the *nidcpower.Session.lcr_load_compensation_enabled* property to True.

Load compensation data are generated only for those specific frequencies you define with this method; load compensation is not interpolated from the specific frequencies you define and applied to other frequencies.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].perform_lcr_load_compensation()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.perform_lcr_load_compensation()`

Parameters

compensation_spots (*list of LCRLoadCompensationSpot*) – Defines the frequencies and DUT specifications to use for LCR load compensation.

You can specify <=1000 spot frequencies.

frequency	The spot frequency, in Hz.
reference_value_t	A known specification value of your DUT to use as the basis for load compensation.
reference_value	A value that describes the **reference_value_type** specification. Use as indicated by the **reference_value_type** option you choose.

perform_lcr_open_compensation

nidcpower.Session.**perform_lcr_open_compensation**(*additional_frequencies=None*)

Generates open compensation data for LCR measurements based on a default set of test frequencies and, optionally, additional frequencies you can specify.

You must physically configure an open LCR circuit to use this method to generate valid open compensation data.

When you call this method:

- The open compensation data is written to the onboard storage of the instrument. Onboard storage can contain only the most recent set of data.
- Most NI-DCPower properties in the session are reset to their default values. Rewrite the values of any properties you want to maintain.

To apply the open compensation data you generate with this method to your LCR measurements, set the *nidcpower.Session.lcr_open_compensation_enabled* property to True.

Corrections for frequencies other than the default frequencies or any additional frequencies you specify are interpolated.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Note: Default Open Compensation Frequencies: By default, NI-DCPower uses the following frequencies for LCR open compensation:

- 10 logarithmic steps at 1 kHz frequency decade
- 10 logarithmic steps at 10 kHz frequency decade
- 100 logarithmic steps at 100 kHz frequency decade
- 100 logarithmic steps at 1 MHz frequency decade

The actual frequencies used depend on the bandwidth of your instrument.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].perform_lcr_open_compensation()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.perform_lcr_open_compensation()`

Parameters

additional_frequencies (*list of float*) – Defines a further set of frequencies, in addition to the default frequencies, to perform the compensation for. You can specify <=200 additional frequencies.

perform_lcr_open_custom_cable_compensation

nidcpower.Session.**perform_lcr_open_custom_cable_compensation**()

Generates open custom cable compensation data for LCR measurements.

To use this method, you must physically configure an open LCR circuit to generate valid open custom cable compensation data.

When you call this method:

- The open compensation data is written to the onboard storage of the instrument. Onboard storage can contain only the most recent set of data.
- Most NI-DCPower properties in the session are reset to their default values. Rewrite the values of any properties you want to maintain.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].perform_lcr_open_custom_cable_compensation()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.perform_lcr_open_custom_cable_compensation()`

perform_lcr_short_compensation

nidcpower.Session.**perform_lcr_short_compensation**(*additional_frequencies=None*)

Generates short compensation data for LCR measurements based on a default set of test frequencies and, optionally, additional frequencies you can specify.

You must physically configure your LCR circuit with a short to use this method to generate valid short compensation data.

When you call this method:

- The short compensation data is written to the onboard storage of the instrument. Onboard storage can contain only the most recent set of data.
- Most NI-DCPower properties in the session are reset to their default values. Rewrite the values of any properties you want to maintain.

To apply the short compensation data you generate with this method to your LCR measurements, set the *nidcpower.Session.lcr_short_compensation_enabled* property to True.

Corrections for frequencies other than the default frequencies or any additional frequencies you specify are interpolated.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Note: Default Short Compensation Frequencies: By default, NI-DCPower uses the following frequencies for LCR short compensation:

- 10 logarithmic steps at 1 kHz frequency decade
- 10 logarithmic steps at 10 kHz frequency decade
- 100 logarithmic steps at 100 kHz frequency decade
- 100 logarithmic steps at 1 MHz frequency decade

The actual frequencies used depend on the bandwidth of your instrument.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].perform_lcr_short_compensation()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.perform_lcr_short_compensation()`

Parameters

additional_frequencies (*list of float*) – Defines a further set of frequencies, in addition to the default frequencies, to perform the compensation for. You can specify <=200 additional frequencies.

perform_lcr_short_custom_cable_compensation

`nidcpower.Session.`**`perform_lcr_short_custom_cable_compensation`**`()`

Generates short custom cable compensation data for LCR measurements.

To use this method:

- You must physically configure your LCR circuit with a short to generate valid short custom cable compensation data.
- Set *nidcpower.Session.lcr_short_custom_cable_compensation_enabled* property to True

When you call this method:

- The short compensation data is written to the onboard storage of the instrument. Onboard storage can contain only the most recent set of data.
- Most NI-DCPower properties in the session are reset to their default values. Rewrite the values of any properties you want to maintain.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].perform_lcr_short_custom_cable_compensation()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.perform_lcr_short_custom_cable_compensation()`

---

query_in_compliance

`nidcpower.Session.`**`query_in_compliance`**`()`

Queries the specified output device to determine if it is operating at the compliance limit.

The compliance limit is the current limit when the output method is set to *DC_VOLTAGE*. If the output is operating at the compliance limit, the output reaches the current limit before the desired voltage level. Refer to the `nidcpower.Session.ConfigureOutputFunction()` method and the `nidcpower.Session.ConfigureCurrentLimit()` method for more information about output method and current limit, respectively.

The compliance limit is the voltage limit when the output method is set to *DC_CURRENT*. If the output is operating at the compliance limit, the output reaches the voltage limit before the desired current level. Refer to the `nidcpower.Session.ConfigureOutputFunction()` method and the `nidcpower.Session.ConfigureVoltageLimit()` method for more information about output method and voltage limit, respectively.

Related Topics:

Compliance

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Note: One or more of the referenced methods are not in the Python API for this driver.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].query_in_compliance()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.query_in_compliance()`

Return type
bool

Returns
Returns whether the device channel is in compliance.

query_latched_output_cutoff_state

`nidcpower.Session.`**`query_latched_output_cutoff_state`**(*output_cutoff_reason*)

Discovers if an output cutoff limit was exceeded for the specified reason. When an output cutoff is engaged, the output of the channel(s) is disconnected. If a limit was exceeded, the state is latched until you clear it with the *nidcpower.Session.clear_latched_output_cutoff_state()* method or the *nidcpower.Session.reset()* method.

outputCutoffReason specifies the conditions for which an output is disconnected.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].query_latched_output_cutoff_state()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.query_latched_output_cutoff_state()`

Parameters

output_cutoff_reason (*nidcpower.OutputCutoffReason*) – Specifies which output cutoff conditions to query.

ALL	Any output cutoff condition was met
VOLTAGE_OUTPUT_H	The output exceeded the high cutoff limit for voltage output
VOLTAGE_OUTPUT_L	The output fell below the low cutoff limit for voltage output
*VOLTAGE_MEASURE_*	The measured voltage exceeded the high cutoff limit for voltage output
*VOLTAGE_MEASURE_*	The measured voltage fell below the low cutoff limit for voltage output
*CURRENT_MEASURE_*	The measured current exceeded the high cutoff limit for current output
*CURRENT_MEASURE_*	The measured current fell below the low cutoff limit for current output
VOLTAGE_CHANGE_H	The voltage slew rate increased beyond the positive change cutoff for voltage output
VOLTAGE_CHANGE_L	The voltage slew rate decreased beyond the negative change cutoff for voltage output
CURRENT_CHANGE_H	The current slew rate increased beyond the positive change cutoff for current output
CURRENT_CHANGE_L	The current slew rate decreased beyond the negative change cutoff for current output
CURRENT_SATURATI	The measured current saturates the current range

Return type

bool

Returns

Specifies whether an output cutoff has engaged.

True	An output cutoff has engaged for the conditions in **output cutoff reason**.
False	No output cutoff has engaged.

query_max_current_limit

`nidcpower.Session.`**`query_max_current_limit`**(*voltage_level*)

Queries the maximum current limit on a channel if the channel is set to the specified **voltageLevel**.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].query_max_current_limit()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.query_max_current_limit()`

Parameters
voltage_level (*float*) – Specifies the voltage level to use when calculating the **maxCurrentLimit**.

Return type
float

Returns
Returns the maximum current limit that can be set with the specified **voltageLevel**.

query_max_voltage_level

nidcpower.Session.**query_max_voltage_level**(*current_limit*)

Queries the maximum voltage level on a channel if the channel is set to the specified **currentLimit**.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].query_max_voltage_level()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.query_max_voltage_level()`

Parameters
current_limit (*float*) – Specifies the current limit to use when calculating the **maxVoltageLevel**.

Return type
float

Returns
Returns the maximum voltage level that can be set on a channel with the specified **currentLimit**.

query_min_current_limit

nidcpower.Session.**query_min_current_limit**(*voltage_level*)

Queries the minimum current limit on a channel if the channel is set to the specified **voltageLevel**.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].query_min_current_limit()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.query_min_current_limit()`

Parameters
voltage_level (*float*) – Specifies the voltage level to use when calculating the **minCurrentLimit**.

Return type
float

Returns
Returns the minimum current limit that can be set on a channel with the specified **voltageLevel**.

query_output_state

nidcpower.Session.**query_output_state**(*output_state*)

Queries the specified channel to determine if the channel is currently in the state specified by **outputState**.

Related Topics:

Compliance

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].query_output_state()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.query_output_state()`

Parameters
output_state (*nidcpower.OutputStates*) – Specifies the output state of the channel that is being queried. **Defined Values**:

VOLTAGE	The device maintains a constant voltage by adjusting the current.
CURRENT	The device maintains a constant current by adjusting the voltage.

Return type
bool

Returns
Returns whether the device channel is in the specified output state.

read_current_temperature

nidcpower.Session.**read_current_temperature**()

Returns the current onboard **temperature**, in degrees Celsius, of the device.

Return type
float

Returns
Returns the onboard **temperature**, in degrees Celsius, of the device.

reset

nidcpower.Session.**reset**()

Resets the specified channel(s) to a known state. This method disables power generation, resets session properties to their default values, commits the session properties, and leaves the session in the Uncommitted state. Refer to the Programming States topic for more information about NI-DCPower software states.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].reset()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.reset()`

---

reset_device

nidcpower.Session.**reset_device**()

Resets the device to a known state. The method disables power generation, resets session properties to their default values, clears errors such as overtemperature and unexpected loss of auxiliary power, commits the session properties, and leaves the session in the Uncommitted state. This method also performs a hard reset on the device and driver software. This method has the same functionality as using reset in Measurement & Automation Explorer. Refer to the Programming States topic for more information about NI-DCPower software states.

This will also open the output relay on devices that have an output relay.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

---

reset_with_defaults

nidcpower.Session.**reset_with_defaults**()

Resets the device to a known state. This method disables power generation, resets session properties to their default values, commits the session properties, and leaves the session in the Running state. In addition to exhibiting the behavior of the *nidcpower.Session.reset()* method, this method can assign user-defined default values for configurable properties from the IVI configuration.

self_cal

nidcpower.Session.**self_cal**()

Performs a self-calibration upon the specified channel(s).

This method disables the output, performs several internal calculations, and updates calibration values. The updated calibration values are written to the device hardware if the *nidcpower.Session.self_calibration_persistence* property is set to *WRITE_TO_EEPROM*. Refer to the *nidcpower.Session.self_calibration_persistence* property topic for more information about the settings for this property.

When calling *nidcpower.Session.self_cal()* with the PXIe-4162/4163, specify all channels of your PXIe-4162/4163 with the channelName input. You cannot self-calibrate a subset of PXIe-4162/4163 channels.

Refer to the Self-Calibration topic for more information about this method.

Related Topics:

Self-Calibration

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].self_cal()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.self_cal()`

---

self_test

nidcpower.Session.**self_test**()

Performs the device self-test routine and returns the test result(s). Calling this method implicitly calls the *nidcpower.Session.reset()* method.

When calling *nidcpower.Session.self_test()* with the PXIe-4162/4163, specify all channels of your PXIe-4162/4163 with the channels input of `nidcpower.Session.__init__()`. You cannot self test a subset of PXIe-4162/4163 channels.

Raises *SelfTestError* on self test failure. Properties on exception object:

- code - failure code from driver
- message - status message from driver

Self-Test Code	Description
0	Self test passed.
1	Self test failed.

send_software_edge_trigger

nidcpower.Session.**send_software_edge_trigger**(*trigger*)

Asserts the specified trigger. This method can override an external edge trigger.

Related Topics:

Triggers

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].send_software_edge_trigger()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.send_software_edge_trigger()`

Parameters

trigger (*nidcpower.SendSoftwareEdgeTriggerType*) – Specifies which trigger to assert. **Defined Values:**

START	Asserts the Start trigger.
SOURCE	Asserts the Source trigger.
MEASURE	Asserts the Measure trigger.
SEQUENCE_ADVANCE	Asserts the Sequence Advance trigger.
PULSE	Asserts the Pulse trigger.
SHUTDOWN	Asserts the Shutdown trigger.

Note: One or more of the referenced values are not in the Python API for this driver. Enums that only define values, or represent True/False, have been removed.

set_sequence

nidcpower.Session.**set_sequence**(*values*, *source_delays*)

Configures a series of voltage or current outputs and corresponding source delays. The source mode must be set to Sequence for this method to take effect.

Refer to the Configuring the Source Unit topic in the *NI DC Power Supplies and SMUs Help* for more information about how to configure your device.

Use this method in the Uncommitted or Committed programming states. Refer to the Programming States topic in the *NI DC Power Supplies and SMUs Help* for more information about NI-DCPower programming states.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].set_sequence()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.set_sequence()`

Parameters

- **values** (*list of float*) – Specifies the series of voltage levels or current levels, depending on the configured output method. **Valid values**: The valid values for this parameter are defined by the voltage level range or current level range.
- **source_delays** (*list of float*) – Specifies the source delay that follows the configuration of each value in the sequence. **Valid Values**: The valid values are between 0 and 167 seconds.

unlock

nidcpower.Session.**unlock**()

Releases a lock that you acquired on an device session using *nidcpower.Session.lock()*. Refer to *nidcpower.Session.unlock()* for additional information on session locks.

wait_for_event

nidcpower.Session.**wait_for_event**(*event_id*, *timeout=hightime.timedelta(seconds=10.0)*)

Waits until the specified channel(s) have generated the specified event.

The session monitors whether each type of event has occurred at least once since the last time this method or the *nidcpower.Session.initiate()* method were called. If an event has only been generated once and you call this method successively, the method times out. Individual events must be generated between separate calls of this method.

Note: This method is not supported on all devices. For more information about supported devices, search ni.com for Supported Methods by Device.

Tip: This method can be called on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset, and then call this method on the result.

Example: `my_session.channels[ ... ].wait_for_event()`

To call the method on all channels, you can call it directly on the *nidcpower.Session*.

Example: `my_session.wait_for_event()`

Parameters

- **event_id** (*nidcpower.Event*) – Specifies which event to wait for. **Defined Values:**

SOURCE_COMPLETE	Waits for the Source Complete event.
MEASURE_COMPLETE	Waits for the Measure Complete event.
SEQUENCE_ITERATION_COMPLET	Waits for the Sequence Iteration Complete event.
SEQUENCE_ENGINE_DONE	Waits for the Sequence Engine Done event.
PULSE_COMPLETE	Waits for the Pulse Complete event.
READY_FOR_PULSE_TRIGGER	Waits for the Ready for Pulse Trigger event.

Note: One or more of the referenced values are not in the Python API for this driver. Enums that only define values, or represent True/False, have been removed.

- **timeout** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Specifies the maximum time allowed for this method to complete, in seconds. If the method does not complete within this time interval, NI-DCPower returns an error.

Note: When setting the timeout interval, ensure you take into account any triggers so that the timeout interval is long enough for your application.

Properties

active_advanced_sequence

nidcpower.Session.**active_advanced_sequence**

Specifies the advanced sequence to configure or generate.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].active_advanced_sequence`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.active_advanced_sequence`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Active Advanced Sequence**
- C Attribute: **NIDCPOWER_ATTR_ACTIVE_ADVANCED_SEQUENCE**

active_advanced_sequence_step

nidcpower.Session.**active_advanced_sequence_step**

Specifies the advanced sequence step to configure.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].active_advanced_sequence_step`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.active_advanced_sequence_step`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Active Advanced Sequence Step**
- C Attribute: **NIDCPOWER_ATTR_ACTIVE_ADVANCED_SEQUENCE_STEP**

actual_power_allocation

`nidcpower.Session.`**`actual_power_allocation`**

Returns the power, in watts, the device is sourcing on each active channel if the *nidcpower.Session.power_allocation_mode* property is set to *AUTOMATIC* or *MANUAL*.

Valid Values: [0, device per-channel maximum power]

Default Value: Refer to the Supported Properties by Device topic for the default value by device.

Note: NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

This property returns -1 when the *nidcpower.Session.power_allocation_mode* property is set to *DISABLED*.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].actual_power_allocation`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.actual_power_allocation`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read only
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Actual Power Allocation**
- C Attribute: **NIDCPOWER_ATTR_ACTUAL_POWER_ALLOCATION**

aperture_time

nidcpower.Session.**aperture_time**

Specifies the measurement aperture time for the channel configuration. Aperture time is specified in the units set by the *nidcpower.Session.aperture_time_units* property. Refer to the Aperture Time topic in the NI DC Power Supplies and SMUs Help for more information about how to configure your measurements and for information about valid values. Default Value: 0.01666666 seconds

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].aperture_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.aperture_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Aperture Time**
- C Attribute: **NIDCPOWER_ATTR_APERTURE_TIME**

aperture_time_auto_mode

nidcpower.Session.**aperture_time_auto_mode**

Automatically optimizes the measurement aperture time according to the actual current range when measurement autorange is enabled. Optimization accounts for power line frequency when the *nidcpower.Session.aperture_time_units* property is set to *POWER_LINE_CYCLES*.

This property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE* and the *nidcpower.Session.autorange* property is enabled.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].aperture_time_auto_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.aperture_time_auto_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ApertureTimeAutoMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Aperture Time Auto Mode**
- C Attribute: **NIDCPOWER_ATTR_APERTURE_TIME_AUTO_MODE**

---

aperture_time_units

nidcpower.Session.**aperture_time_units**

Specifies the units of the *nidcpower.Session.aperture_time* property for the channel configuration. Refer to the Aperture Time topic in the NI DC Power Supplies and SMUs Help for more information about how to configure your measurements and for information about valid values. Default Value: *SECONDS*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].aperture_time_units`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.aperture_time_units`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ApertureTimeUnits
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Aperture Time Units**
- C Attribute: **NIDCPOWER_ATTR_APERTURE_TIME_UNITS**

autorange

`nidcpower.Session.autorange`

Specifies whether the hardware automatically selects the best range to measure the signal. Note the highest range the algorithm uses is dependent on the corresponding limit range property. The algorithm the hardware uses can be controlled using the *nidcpower.Session.autorange_aperture_time_mode* property.

Note: Autoranging begins at module startup and remains active until the module is reconfigured or reset. This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Autorange**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE**

autorange_aperture_time_mode

nidcpower.Session.**autorange_aperture_time_mode**

Specifies whether the aperture time used for the measurement autorange algorithm is determined automatically or customized using the *nidcpower.Session.autorange_minimum_aperture_time* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_aperture_time_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_aperture_time_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.AutorangeApertureTimeMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Aperture Time Mode**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_APERTURE_TIME_MODE**

autorange_behavior

nidcpower.Session.**autorange_behavior**

Specifies the algorithm the hardware uses for measurement autoranging.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_behavior`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_behavior`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.AutorangeBehavior
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Behavior**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_BEHAVIOR**

autorange_maximum_delay_after_range_change

nidcpower.Session.**autorange_maximum_delay_after_range_change**

Balances between settling time and maximum measurement time by specifying the maximum time delay between when a range change occurs and when measurements resume. **Valid Values:** The minimum and maximum values of this property are hardware-dependent. PXIe-4135/4136/4137: 0 to 9 seconds PXIe-4138/4139: 0 to 9 seconds PXIe-4147: 0 to 9 seconds PXIe-4162/4163: 0 to 0.1 seconds.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_maximum_delay_after_range_change`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_maximum_delay_after_range_change`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Maximum Delay After Range Change**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_MAXIMUM_DELAY_AFTER_RANGE_CHANGE**

autorange_minimum_aperture_time

nidcpower.Session.**autorange_minimum_aperture_time**

Specifies the measurement autorange aperture time used for the measurement autorange algorithm. The aperture time is specified in the units set by the *nidcpower.Session.autorange_minimum_aperture_time_units* property. This value will typically be smaller than the aperture time used for measurements.

Note: For smaller ranges, the value is scaled up to account for noise. The factor used to scale the value is derived from the module capabilities. This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_minimum_aperture_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_minimum_aperture_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Minimum Aperture Time**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_MINIMUM_APERTURE_TIME**

autorange_minimum_aperture_time_units

nidcpower.Session.**autorange_minimum_aperture_time_units**

Specifies the units of the *nidcpower.Session.autorange_minimum_aperture_time* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_minimum_aperture_time_units`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_minimum_aperture_time_units`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ApertureTimeUnits
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Minimum Aperture Time Units**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_MINIMUM_APERTURE_TIME_UNITS**

autorange_minimum_current_range

nidcpower.Session.**autorange_minimum_current_range**

Specifies the lowest range used during measurement autoranging. Limiting the lowest range used during autoranging can improve the speed of the autoranging algorithm and minimize frequent and unpredictable range changes for noisy signals.

Note: The maximum range used is the range that includes the value specified in the compliance limit property, *nidcpower.Session.voltage_limit_range* property or *nidcpower.Session.current_limit_range* property, depending on the selected *nidcpower.Session.output_function*. This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_minimum_current_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_minimum_current_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Minimum Current Range**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_MINIMUM_CURRENT_RANGE**

autorange_minimum_voltage_range

`nidcpower.Session.`**`autorange_minimum_voltage_range`**

Specifies the lowest range used during measurement autoranging. The maximum range used is range that includes the value specified in the compliance limit property. Limiting the lowest range used during autoranging can improve the speed of the autoranging algorithm and/or minimize thrashing between ranges for noisy signals.

Note: The maximum range used is the range that includes the value specified in the compliance limit property, *nidcpower.Session.voltage_limit_range* property or *nidcpower.Session.current_limit_range* property, depending on the selected *nidcpower.Session.output_function*. This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_minimum_voltage_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_minimum_voltage_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Minimum Voltage Range**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_MINIMUM_VOLTAGE_RANGE**

autorange_threshold_mode

nidcpower.Session.**autorange_threshold_mode**

Specifies thresholds used during autoranging to determine when range changing occurs.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].autorange_threshold_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.autorange_threshold_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.AutorangeThresholdMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Autorange Threshold Mode**
- C Attribute: **NIDCPOWER_ATTR_AUTORANGE_THRESHOLD_MODE**

auto_zero

nidcpower.Session.**auto_zero**

Specifies the auto-zero method to use on the device. Refer to the NI PXI-4132 Measurement Configuration and Timing and Auto Zero topics for more information about how to configure your measurements. Default Value: The default value for the NI PXI-4132 is *ON*. The default value for all other devices is *OFF*, which is the only supported value for these devices.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].auto_zero`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.auto_zero`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.AutoZero
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Auto Zero**
- C Attribute: **NIDCPOWER_ATTR_AUTO_ZERO**

auxiliary_power_source_available

nidcpower.Session.**auxiliary_power_source_available**

Indicates whether an auxiliary power source is connected to the device. A value of False may indicate that the auxiliary input fuse has blown. Refer to the Detecting Internal/Auxiliary Power topic in the NI DC Power Supplies and SMUs Help for more information about internal and auxiliary power. power source to generate power. Use the *nidcpower.Session.power_source_in_use* property to retrieve this information.

Note: This property does not necessarily indicate if the device is using the auxiliary

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Advanced:Auxiliary Power Source Available**
- C Attribute: **NIDCPOWER_ATTR_AUXILIARY_POWER_SOURCE_AVAILABLE**

cable_length

nidcpower.Session.**cable_length**

Specifies how to apply cable compensation data for instruments that support LCR functionality. Supported instruments use cable compensation for the following operations:

SMU mode: to stabilize DC current sourcing in the two smallest current ranges. LCR mode: to compensate for the effects of cabling on LCR measurements.

For NI standard options, select the length of your NI cable to apply compensation data for a typical cable of that type. For custom options, choose the source of the custom cable compensation data. You must then generate the custom cable compensation data.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].cable_length`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.cable_length`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.CableLength
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Device Specific:LCR:Cable Length**
- C Attribute: **NIDCPOWER_ATTR_CABLE_LENGTH**

---

channel_count

nidcpower.Session.**channel_count**

Indicates the number of channels that NI-DCPower supports for the instrument that was chosen when the current session was opened. For channel-based properties, the IVI engine maintains a separate cache value for each channel.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Driver Capabilities:Channel Count**
- C Attribute: **NIDCPOWER_ATTR_CHANNEL_COUNT**

compliance_limit_symmetry

nidcpower.Session.**compliance_limit_symmetry**

Specifies whether compliance limits for current generation and voltage generation for the device are applied symmetrically about 0 V and 0 A or asymmetrically with respect to 0 V and 0 A. When set to *SYMMETRIC*, voltage limits and current limits are set using a single property with a positive value. The resulting range is bounded by this positive value and its opposite. When set to *ASYMMETRIC*, you must separately set a limit high and a limit low using distinct properties. For asymmetric limits, the range bounded by the limit high and limit low must include zero. **Default Value:** Symmetric **Related Topics:** Compliance; Ranges; Changing Ranges; Overranging

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].compliance_limit_symmetry`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.compliance_limit_symmetry`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ComplianceLimitSymmetry
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Compliance Limit Symmetry**
- C Attribute: **NIDCPOWER_ATTR_COMPLIANCE_LIMIT_SYMMETRY**

conduction_voltage_mode

nidcpower.Session.**conduction_voltage_mode**

Specifies whether the conduction voltage feature is enabled on the specified channel(s).

When the conduction voltage feature is enabled,

- The instrument will not begin sinking on the specified channel(s) until the voltage at the input of the specified channel(s) rises above *nidcpower.Session.conduction_voltage_on_threshold*
- The instrument will stop sinking if the voltage at the input of the specified channel(s) falls below *nidcpower.Session.conduction_voltage_off_threshold*.

When the conduction voltage feature is disabled,

- The instrument will start sinking regardless of the voltage at the input of the specified channel(s).

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].conduction_voltage_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.conduction_voltage_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ConductionVoltageMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Conduction Voltage:Mode**
- C Attribute: **NIDCPOWER_ATTR_CONDUCTION_VOLTAGE_MODE**

---

conduction_voltage_off_threshold

nidcpower.Session.**conduction_voltage_off_threshold**

Specifies the minimum voltage, in volts, at the input of the specified channel(s) below which the instrument stops sinking on the specified channel(s) when the conduction voltage feature is enabled.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].conduction_voltage_off_threshold`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.conduction_voltage_off_threshold`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Conduction Voltage:Off Threshold**
- C Attribute: **NIDCPOWER_ATTR_CONDUCTION_VOLTAGE_OFF_THRESHOLD**

---

conduction_voltage_on_threshold

nidcpower.Session.**conduction_voltage_on_threshold**

Specifies the required minimum voltage, in volts, at the input of the specified channel(s) before the instrument starts sinking on the specified channel(s) when the conduction voltage feature is enabled.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].conduction_voltage_on_threshold`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.conduction_voltage_on_threshold`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Conduction Voltage:On Threshold**
- C Attribute: **NIDCPOWER_ATTR_CONDUCTION_VOLTAGE_ON_THRESHOLD**

current_compensation_frequency

`nidcpower.Session.`**`current_compensation_frequency`**

The frequency at which a pole-zero pair is added to the system when the channel is in Constant Current mode. Default Value: Determined by the value of the *NORMAL* setting of the *nidcpower.Session.transient_response* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_compensation_frequency`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_compensation_frequency`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Custom Transient Response:Current:Compensation Frequency**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_COMPENSATION_FREQUENCY**

current_gain_bandwidth

nidcpower.Session.**current_gain_bandwidth**

The frequency at which the unloaded loop gain extrapolates to 0 dB in the absence of additional poles and zeroes. This property takes effect when the channel is in Constant Current mode. Default Value: Determined by the value of the *NORMAL* setting of the *nidcpower.Session.transient_response* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_gain_bandwidth`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_gain_bandwidth`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Custom Transient Response:Current:Gain Bandwidth**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_GAIN_BANDWIDTH**

current_level

nidcpower.Session.**current_level**

Specifies the current level, in amps, that the device attempts to generate on the specified channel(s). This property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_CURRENT*.

Valid Values: The valid values for this property are defined by the values to which the *nidcpower.Session.current_level_range* property is set.

Note: The channel must be enabled for the specified current level to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Current Level**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LEVEL**

current_level_autorange

nidcpower.Session.**current_level_autorange**

Specifies whether NI-DCPower automatically selects the current level range based on the desired current level for the specified channels. If you set this property to *ON*, NI-DCPower ignores any changes you make to the *nidcpower.Session.current_level_range* property. If you change the *nidcpower.Session.current_level_autorange* property from *ON* to *OFF*, NI-DCPower retains the last value the *nidcpower.Session.current_level_range* property was set to (or the default value if the property was never set) and uses that value as the current level range. Query the *nidcpower.Session.current_level_range* property by using the `nidcpower.Session._get_attribute_vi_int32()` method for information about which range NI-DCPower automatically selects. The *nidcpower.Session.current_level_autorange* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_CURRENT*. Default Value: *OFF*

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_level_autorange`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_level_autorange`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Current Level Autorange**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LEVEL_AUTORANGE**

current_level_falling_slew_rate

nidcpower.Session.**current_level_falling_slew_rate**

Specifies the rate of decrease, in amps per microsecond, to apply to the absolute magnitude of the current level of the specified channel(s). This property is applicable only if you set the *nidcpower.Session.output_function* property to *DC_CURRENT*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_level_falling_slew_rate`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_level_falling_slew_rate`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Current Level Slew Rate:Falling**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LEVEL_FALLING_SLEW_RATE**

current_level_range

nidcpower.Session.**current_level_range**

Specifies the current level range, in amps, for the specified channel(s). The range defines the valid values to which you can set the current level. Use the *nidcpower.Session.current_level_autorange* property to enable automatic selection of the current level range. The *nidcpower.Session.current_level_range* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_CURRENT*.

For valid ranges, refer to the specifications for your instrument.

Note: The channel must be enabled for the specified current level range to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_level_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_level_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Current Level Range**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE**

---

current_level_rising_slew_rate

nidcpower.Session.**current_level_rising_slew_rate**

Specifies the rate of increase, in amps per microsecond, to apply to the absolute magnitude of the current level of the specified channel(s). This property is applicable only if you set the *nidcpower.Session.output_function* property to *DC_CURRENT*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_level_rising_slew_rate`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_level_rising_slew_rate`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Current Level Slew Rate:Rising**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LEVEL_RISING_SLEW_RATE**

current_limit

`nidcpower.Session.`**`current_limit`**

Specifies the current limit, in amps, that the output cannot exceed when generating the desired voltage level on the specified channel(s). This property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE* and the *nidcpower.Session.compliance_limit_symmetry* property is set to *SYMMETRIC*.

Valid Values: The valid values for this property are defined by the values to which *nidcpower.Session.current_limit_range* property is set.

Note: The channel must be enabled for the specified current limit to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_limit`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_limit`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Current Limit**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LIMIT**

current_limit_autorange

nidcpower.Session.**current_limit_autorange**

Specifies whether NI-DCPower automatically selects the current limit range based on the desired current limit for the specified channel(s). If you set this property to *ON*, NI-DCPower ignores any changes you make to the *nidcpower.Session.current_limit_range* property. If you change this property from *ON* to *OFF*, NI-DCPower retains the last value the *nidcpower.Session.current_limit_range* property was set to (or the default value if the property was never set) and uses that value as the current limit range. Query the *nidcpower.Session.current_limit_range* property by using the `nidcpower.Session._get_attribute_vi_int32()` method for information about which range NI-DCPower automatically selects. The *nidcpower.Session.current_limit_autorange* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE*. Default Value: *OFF*

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_limit_autorange`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_limit_autorange`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Current Limit Autorange**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LIMIT_AUTORANGE**

current_limit_behavior

nidcpower.Session.**current_limit_behavior**

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_limit_behavior`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_limit_behavior`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.CurrentLimitBehavior
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- C Attribute: **NIDCPOWER_ATTR_CURRENT_LIMIT_BEHAVIOR**

---

current_limit_high

nidcpower.Session.**current_limit_high**

Specifies the maximum current, in amps, that the output can produce when generating the desired voltage on the specified channel(s). This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE*. You must also specify a *nidcpower.Session.current_limit_low* to complete the asymmetric range. **Valid Values:** [1% of *nidcpower.Session.current_limit_range*, *nidcpower.Session.current_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

---

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True.

NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Current Limit High**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LIMIT_HIGH**

current_limit_low

`nidcpower.Session.`**`current_limit_low`**

Specifies the minimum current, in amps, that the output can produce when generating the desired voltage on the specified channel(s). This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE*. You must also specify a *nidcpower.Session.current_limit_high* to complete the asymmetric range. **Valid Values:** [-*nidcpower.Session.current_limit_range*, -1% of *nidcpower.Session.current_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True.

NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Current Limit Low**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LIMIT_LOW**

current_limit_range

nidcpower.Session.**current_limit_range**

Specifies the current limit range, in amps, for the specified channel(s). The range defines the valid values to which you can set the current limit. Use the *nidcpower.Session.current_limit_autorange* property to enable automatic selection of the current limit range. The *nidcpower.Session.current_limit_range* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE*.

For valid ranges, refer to the specifications for your instrument.

Note: The channel must be enabled for the specified current limit to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_limit_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_limit_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Current Limit Range**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE**

current_pole_zero_ratio

nidcpower.Session.**current_pole_zero_ratio**

The ratio of the pole frequency to the zero frequency when the channel is in Constant Current mode. Default Value: Determined by the value of the *NORMAL* setting of the *nidcpower.Session.transient_response* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].current_pole_zero_ratio`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.current_pole_zero_ratio`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Custom Transient Response:Current:Pole-Zero Ratio**
- C Attribute: **NIDCPOWER_ATTR_CURRENT_POLE_ZERO_RATIO**

dc_noise_rejection

nidcpower.Session.**dc_noise_rejection**

Determines the relative weighting of samples in a measurement. Refer to the NI PXIe-4140/4141 DC Noise Rejection, NI PXIe-4142/4143 DC Noise Rejection, or NI PXIe-4144/4145 DC Noise Rejection topic in the NI DC Power Supplies and SMUs Help for more information about noise rejection. Default Value: *NORMAL*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].dc_noise_rejection`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.dc_noise_rejection`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.DCNoiseRejection
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:DC Noise Rejection**
- C Attribute: **NIDCPOWER_ATTR_DC_NOISE_REJECTION**

digital_edge_measure_trigger_input_terminal

nidcpower.Session.**digital_edge_measure_trigger_input_terminal**

Specifies the input terminal for the Measure trigger. This property is used only when the *nidcpower.Session.measure_trigger_type* property is set to *DIGITAL_EDGE*. for this property. You can specify any valid input terminal for this property. Valid terminals are listed in Measurement & Automation Explorer under the Device Routes tab. Input terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0. The input terminal can also be a terminal from another device. For example, you can set the input terminal on Dev1 to be /Dev2/SourceCompleteEvent.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].digital_edge_measure_trigger_input_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.digital_edge_measure_trigger_input_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Measure Trigger:Digital Edge:Input Terminal**
- C Attribute: **NIDCPOWER_ATTR_DIGITAL_EDGE_MEASURE_TRIGGER_INPUT_TERMINAL**

digital_edge_pulse_trigger_input_terminal

nidcpower.Session.**digital_edge_pulse_trigger_input_terminal**

Specifies the input terminal for the Pulse trigger. This property is used only when the *nidcpower.Session.pulse_trigger_type* property is set to digital edge. You can specify any valid input terminal for this property. Valid terminals are listed in Measurement & Automation Explorer under the Device Routes tab. Input terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0. The input terminal can also be a terminal from another device. For example, you can set the input terminal on Dev1 to be /Dev2/SourceCompleteEvent.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].digital_edge_pulse_trigger_input_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.digital_edge_pulse_trigger_input_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Pulse Trigger:Digital Edge:Input Terminal**
- C Attribute: **NIDCPOWER_ATTR_DIGITAL_EDGE_PULSE_TRIGGER_INPUT_TERMINAL**

digital_edge_sequence_advance_trigger_input_terminal

nidcpower.Session.**digital_edge_sequence_advance_trigger_input_terminal**

Specifies the input terminal for the Sequence Advance trigger. Use this property only when the *nidcpower.Session.sequence_advance_trigger_type* property is set to *DIGITAL_EDGE*. the NI DC Power Supplies and SMUs Help for information about supported devices. You can specify any valid input terminal for this property. Valid terminals are listed in Measurement & Automation Explorer under the Device Routes tab. Input terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0. The input terminal can also be a terminal from another device. For example, you can set the input terminal on Dev1 to be /Dev2/SourceCompleteEvent.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].digital_edge_sequence_advance_trigger_input_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.digital_edge_sequence_advance_trigger_input_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Sequence Advance Trigger:Digital Edge:Input Terminal**
- C Attribute: **NIDCPOWER_ATTR_DIGITAL_EDGE_SEQUENCE_ADVANCE_TRIGGER_INPUT_TERMINA**

---

digital_edge_shutdown_trigger_input_terminal

nidcpower.Session.**digital_edge_shutdown_trigger_input_terminal**

Specifies the input terminal for the Shutdown trigger. This property is used only when the *nidcpower.Session.shutdown_trigger_type* property is set to digital edge. You can specify any valid input terminal for this property. Valid terminals are listed in Measurement & Automation Explorer under the Device Routes tab. Input terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0. The input terminal can also be a terminal from another device. For example, you can set the input terminal on Dev1 to be /Dev2/SourceCompleteEvent.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].digital_edge_shutdown_trigger_input_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.digital_edge_shutdown_trigger_input_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Shutdown Trigger:Digital Edge:Input Terminal**
- C Attribute: **NIDCPOWER_ATTR_DIGITAL_EDGE_SHUTDOWN_TRIGGER_INPUT_TERMINAL**

digital_edge_source_trigger_input_terminal

`nidcpower.Session.`**`digital_edge_source_trigger_input_terminal`**

Specifies the input terminal for the Source trigger. Use this property only when the *nidcpower.Session.source_trigger_type* property is set to *DIGITAL_EDGE*. You can specify any valid input terminal for this property. Valid terminals are listed in Measurement & Automation Explorer under the Device Routes tab. Input terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0. The input terminal can also be a terminal from another device. For example, you can set the input terminal on Dev1 to be /Dev2/SourceCompleteEvent.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].digital_edge_source_trigger_input_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.digital_edge_source_trigger_input_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Source Trigger:Digital Edge:Input Terminal**
- C Attribute: **NIDCPOWER_ATTR_DIGITAL_EDGE_SOURCE_TRIGGER_INPUT_TERMINAL**

digital_edge_start_trigger_input_terminal

`nidcpower.Session.`**`digital_edge_start_trigger_input_terminal`**

Specifies the input terminal for the Start trigger. Use this property only when the *nidcpower.Session.start_trigger_type* property is set to *DIGITAL_EDGE*. You can specify any valid input terminal for this property. Valid terminals are listed in Measurement & Automation Explorer under the Device Routes tab. Input terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0. The input terminal can also be a terminal from another device. For example, you can set the input terminal on Dev1 to be /Dev2/SourceCompleteEvent.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].digital_edge_start_trigger_input_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.digital_edge_start_trigger_input_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Start Trigger:Digital Edge:Input Terminal**
- C Attribute: **NIDCPOWER_ATTR_DIGITAL_EDGE_START_TRIGGER_INPUT_TERMINAL**

driver_setup

nidcpower.Session.**driver_setup**

Indicates the Driver Setup string that you specified when initializing the driver. Some cases exist where you must specify the instrument driver options at initialization time. An example of this case is specifying a particular device model from among a family of devices that the driver supports. This property is useful when simulating a device. You can specify the driver-specific options through the DriverSetup keyword in the optionsString parameter in the `nidcpower.Session.__init__()` method or through the IVI Configuration Utility. You can specify driver-specific options through the DriverSetup keyword in the optionsString parameter in the `nidcpower.Session.__init__()` method. If you do not specify a Driver Setup string, this property returns an empty string.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Advanced Session Information:Driver Setup**
- C Attribute: **NIDCPOWER_ATTR_DRIVER_SETUP**

exported_measure_trigger_output_terminal

nidcpower.Session.**exported_measure_trigger_output_terminal**

Specifies the output terminal for exporting the Measure trigger. Refer to the Device Routes tab in Measurement & Automation Explorer for a list of the terminals available on your device. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].exported_measure_trigger_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.exported_measure_trigger_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Measure Trigger:Export Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_EXPORTED_MEASURE_TRIGGER_OUTPUT_TERMINAL**

exported_pulse_trigger_output_terminal

`nidcpower.Session.`**`exported_pulse_trigger_output_terminal`**

Specifies the output terminal for exporting the Pulse trigger. Refer to the Device Routes tab in Measurement & Automation Explorer for a list of the terminals available on your device. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].exported_pulse_trigger_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.exported_pulse_trigger_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Pulse Trigger:Export Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_EXPORTED_PULSE_TRIGGER_OUTPUT_TERMINAL**

exported_sequence_advance_trigger_output_terminal

nidcpower.Session.**exported_sequence_advance_trigger_output_terminal**

Specifies the output terminal for exporting the Sequence Advance trigger. Refer to the Device Routes tab in Measurement & Automation Explorer for a list of the terminals available on your device. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].exported_sequence_advance_trigger_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.exported_sequence_advance_trigger_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Sequence Advance Trigger:Export Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_EXPORTED_SEQUENCE_ADVANCE_TRIGGER_OUTPUT_TERMINAL**

exported_source_trigger_output_terminal

nidcpower.Session.**exported_source_trigger_output_terminal**

Specifies the output terminal for exporting the Source trigger. Refer to the Device Routes tab in MAX for a list of the terminals available on your device. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].exported_source_trigger_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.exported_source_trigger_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Source Trigger:Export Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_EXPORTED_SOURCE_TRIGGER_OUTPUT_TERMINAL**

exported_start_trigger_output_terminal

`nidcpower.Session.`**`exported_start_trigger_output_terminal`**

Specifies the output terminal for exporting the Start trigger. Refer to the Device Routes tab in Measurement & Automation Explorer (MAX) for a list of the terminals available on your device. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].exported_start_trigger_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.exported_start_trigger_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Start Trigger:Export Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_EXPORTED_START_TRIGGER_OUTPUT_TERMINAL**

fetch_backlog

nidcpower.Session.**fetch_backlog**

Returns the number of measurements acquired that have not been fetched yet.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].fetch_backlog`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.fetch_backlog`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Fetch Backlog**
- C Attribute: **NIDCPOWER_ATTR_FETCH_BACKLOG**

instrument_firmware_revision

nidcpower.Session.**instrument_firmware_revision**

Contains the firmware revision information for the device you are currently using.

Tip: This property can be set/get on specific instruments within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container instruments to specify a subset.

Example: `my_session.instruments[ ... ].instrument_firmware_revision`

To set/get on all instruments, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.instrument_firmware_revision`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	instruments

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Firmware Revision**
- C Attribute: **NIDCPOWER_ATTR_INSTRUMENT_FIRMWARE_REVISION**

---

instrument_manufacturer

nidcpower.Session.**instrument_manufacturer**

Contains the name of the manufacturer for the device you are currently using.

Tip: This property can be set/get on specific instruments within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container instruments to specify a subset.

Example: `my_session.instruments[ ... ].instrument_manufacturer`

To set/get on all instruments, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.instrument_manufacturer`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	instruments

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Manufacturer**
- C Attribute: **NIDCPOWER_ATTR_INSTRUMENT_MANUFACTURER**

instrument_mode

nidcpower.Session.**instrument_mode**

Specifies the mode of operation for an instrument channel for instruments that support multiple modes.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].instrument_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.instrument_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.InstrumentMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Instrument Mode**
- C Attribute: **NIDCPOWER_ATTR_INSTRUMENT_MODE**

instrument_model

nidcpower.Session.**instrument_model**

Contains the model number or name of the device that you are currently using.

Tip: This property can be set/get on specific instruments within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container instruments to specify a subset.

Example: `my_session.instruments[ ... ].instrument_model`

To set/get on all instruments, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.instrument_model`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	instruments

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Model**
- C Attribute: **NIDCPOWER_ATTR_INSTRUMENT_MODEL**

interlock_input_open

`nidcpower.Session.`**`interlock_input_open`**

Indicates whether the safety interlock circuit is open. Refer to the Safety Interlock topic in the NI DC Power Supplies and SMUs Help for more information about the safety interlock circuit. about supported devices.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific instruments within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container instruments to specify a subset.

Example: `my_session.instruments[ ... ].interlock_input_open`

To set/get on all instruments, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.interlock_input_open`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read only
Repeated Capabilities	instruments

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Advanced:Interlock Input Open**

- C Attribute: **NIDCPOWER_ATTR_INTERLOCK_INPUT_OPEN**

io_resource_descriptor

nidcpower.Session.**io_resource_descriptor**

Indicates the resource descriptor NI-DCPower uses to identify the physical device. If you initialize NI-DCPower with a logical name, this property contains the resource descriptor that corresponds to the entry in the IVI Configuration utility. If you initialize NI-DCPower with the resource descriptor, this property contains that value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Advanced Session Information:Resource Descriptor**
- C Attribute: **NIDCPOWER_ATTR_IO_RESOURCE_DESCRIPTOR**

isolation_state

nidcpower.Session.**isolation_state**

Defines whether the channel is isolated.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].isolation_state`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.isolation_state`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Advanced:Isolation State**
- C Attribute: **NIDCPOWER_ATTR_ISOLATION_STATE**

lcr_actual_load_reactance

nidcpower.Session.**lcr_actual_load_reactance**

Specifies the actual reactance, in ohms, of the load used for load LCR compensation. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_actual_load_reactance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_actual_load_reactance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:LCR Actual Load Reactance**
- C Attribute: **NIDCPOWER_ATTR_LCR_ACTUAL_LOAD_REACTANCE**

lcr_actual_load_resistance

nidcpower.Session.**lcr_actual_load_resistance**

Specifies the actual resistance, in ohms, of the load used for load LCR compensation. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_actual_load_resistance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_actual_load_resistance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:LCR Actual Load Resistance**
- C Attribute: **NIDCPOWER_ATTR_LCR_ACTUAL_LOAD_RESISTANCE**

lcr_ac_dither_enabled

nidcpower.Session.**lcr_ac_dither_enabled**

Specifies whether dithering is enabled during LCR measurements. Dithering adds out-of-band noise to improve measurements of small voltage and current signals.

Note: Hardware is only warranted to meet its accuracy specs with dither enabled. You can disable dither if the added noise interferes with your device-under-test.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_ac_dither_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_ac_dither_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Advanced:Dither Enabled**
- C Attribute: **NIDCPOWER_ATTR_LCR_AC_DITHER_ENABLED**

lcr_ac_electrical_cable_length_delay

nidcpower.Session.**lcr_ac_electrical_cable_length_delay**

Specifies the one-way electrical length delay of the cable, in seconds. The default value depends on *nidcpower.Session.cable_length*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_ac_electrical_cable_length_delay`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_ac_electrical_cable_length_delay`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:LCR AC Electrical Cable Length Delay**
- C Attribute: **NIDCPOWER_ATTR_LCR_AC_ELECTRICAL_CABLE_LENGTH_DELAY**

lcr_automatic_level_control

nidcpower.Session.**lcr_automatic_level_control**

Specifies whether the channel actively attempts to maintain a constant test voltage or current across the DUT for LCR measurements. The use of voltage or current depends on the test signal you configure with the *nidcpower.Session.lcr_stimulus_function* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_automatic_level_control`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_automatic_level_control`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Automatic Level Control**
- C Attribute: **NIDCPOWER_ATTR_LCR_AUTOMATIC_LEVEL_CONTROL**

lcr_current_amplitude

nidcpower.Session.**lcr_current_amplitude**

Specifies the amplitude, in amps RMS, of the AC current test signal applied to the DUT for LCR measurements. This property applies when the *nidcpower.Session.lcr_stimulus_function* property is set to *CURRENT*.

Valid Values: 7.08e-9 A RMS to 0.707 A RMS

Instrument specifications affect the valid values you can program. Refer to the specifications for your instrument for more information.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_current_amplitude`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_current_amplitude`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Current Amplitude**
- C Attribute: **NIDCPOWER_ATTR_LCR_CURRENT_AMPLITUDE**

lcr_current_range

nidcpower.Session.**lcr_current_range**

Specifies the current range, in amps RMS, for the specified channel(s). The range defines the valid values to which you can set the *nidcpower.Session.lcr_current_amplitude*. For valid ranges, refer to the specifications for your instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_current_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_current_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Advanced:Current Range**
- C Attribute: **NIDCPOWER_ATTR_LCR_CURRENT_RANGE**

lcr_custom_measurement_time

nidcpower.Session.**lcr_custom_measurement_time**

Specifies the LCR measurement aperture time for a channel, in seconds, when the *nidcpower.Session.lcr_measurement_time* property is set to *CUSTOM*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_custom_measurement_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_custom_measurement_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Custom Measurement Time**
- C Attribute: **NIDCPOWER_ATTR_LCR_CUSTOM_MEASUREMENT_TIME**

lcr_dc_bias_automatic_level_control

nidcpower.Session.**lcr_dc_bias_automatic_level_control**

Specifies whether the channel actively maintains a constant DC bias voltage or current across the DUT for LCR measurements. To use this property, you must configure a DC bias by 1) selecting an *nidcpower.Session.lcr_dc_bias_source* and 2) depending on the DC bias source you choose, setting either the *nidcpower.Session.lcr_dc_bias_voltage_level* or *nidcpower.Session.lcr_dc_bias_current_level*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_dc_bias_automatic_level_control`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_dc_bias_automatic_level_control`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:DC Bias:Automatic Level Control**
- C Attribute: **NIDCPOWER_ATTR_LCR_DC_BIAS_AUTOMATIC_LEVEL_CONTROL**

lcr_dc_bias_current_level

`nidcpower.Session.`**`lcr_dc_bias_current_level`**

Specifies the DC bias current level, in amps, when the *nidcpower.Session.lcr_dc_bias_source* property is set to *CURRENT*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_dc_bias_current_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_dc_bias_current_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:DC Bias:Current Level**
- C Attribute: **NIDCPOWER_ATTR_LCR_DC_BIAS_CURRENT_LEVEL**

lcr_dc_bias_current_range

nidcpower.Session.**lcr_dc_bias_current_range**

Specifies the DC Bias current range, in amps, for the specified channel(s). The range defines the valid values to which you can set the *nidcpower.Session.lcr_dc_bias_current_level*. For valid ranges, refer to the specifications for your instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_dc_bias_current_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_dc_bias_current_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:DC Bias:Advanced:Current Range**
- C Attribute: **NIDCPOWER_ATTR_LCR_DC_BIAS_CURRENT_RANGE**

lcr_dc_bias_source

nidcpower.Session.**lcr_dc_bias_source**

Specifies how to apply DC bias for LCR measurements.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_dc_bias_source`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_dc_bias_source`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCRDCBiasSource
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:DC Bias:Source**
- C Attribute: **NIDCPOWER_ATTR_LCR_DC_BIAS_SOURCE**

lcr_dc_bias_transient_response

nidcpower.Session.**lcr_dc_bias_transient_response**

For instruments in LCR mode, determines whether NI-DCPower automatically calculates and applies the transient response values for DC bias or applies the transient response you set manually.

Default Value: Search ni.com for Supported Properties by Device for the default value by instrument.

Related Topics: Transient Response

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_dc_bias_transient_response`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_dc_bias_transient_response`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCRDCBiasTransientResponse
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:DC Bias:Advanced:Transient Response**
- C Attribute: **NIDCPOWER_ATTR_LCR_DC_BIAS_TRANSIENT_RESPONSE**

lcr_dc_bias_voltage_level

`nidcpower.Session.`**`lcr_dc_bias_voltage_level`**

Specifies the DC bias voltage level, in volts, when the *nidcpower.Session.lcr_dc_bias_source* property is set to *VOLTAGE*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_dc_bias_voltage_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_dc_bias_voltage_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:DC Bias:Voltage Level**
- C Attribute: **NIDCPOWER_ATTR_LCR_DC_BIAS_VOLTAGE_LEVEL**

lcr_dc_bias_voltage_range

nidcpower.Session.**lcr_dc_bias_voltage_range**

Specifies the DC Bias voltage range, in volts, for the specified channel(s). The range defines the valid values to which you can set the *nidcpower.Session.lcr_dc_bias_voltage_level*. For valid ranges, refer to the specifications for your instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_dc_bias_voltage_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_dc_bias_voltage_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:DC Bias:Advanced:Voltage Range**
- C Attribute: **NIDCPOWER_ATTR_LCR_DC_BIAS_VOLTAGE_RANGE**

lcr_frequency

nidcpower.Session.**lcr_frequency**

Specifies the frequency of the AC test signal applied to the DUT for LCR measurements.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_frequency`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_frequency`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Frequency**
- C Attribute: **NIDCPOWER_ATTR_LCR_FREQUENCY**

lcr_impedance_auto_range

`nidcpower.Session.lcr_impedance_auto_range`

Defines whether an instrument in LCR mode automatically selects the best impedance range for each given LCR measurement.

Impedance autoranging may be enabled only when both:

- The *nidcpower.Session.source_mode* property is set to *SINGLE_POINT*
- *nidcpower.Session.measure_when* is set to a value other than *ON_MEASURE_TRIGGER*

You can read *nidcpower.Session.lcr_impedance_range* back after a measurement to determine the actual range used.

When enabled, impedance autoranging overrides impedance range settings you configure manually with any other properties.

When using a load with unknown impedance, you can set this property to `ON` to determine the correct impedance range for the load. When you know the load impedance, you can achieve faster performance by setting this property to `OFF` and setting *nidcpower.Session.lcr_impedance_range_source* to *LOAD_CONFIGURATION*.

Default Value: Search ni.com for Supported Properties by Device for the default value by instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_impedance_auto_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_impedance_auto_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Impedance Range:Impedance Autorange**
- C Attribute: **NIDCPOWER_ATTR_LCR_IMPEDANCE_AUTO_RANGE**

lcr_impedance_range

nidcpower.Session.**lcr_impedance_range**

Specifies the impedance range the channel uses for LCR measurements.

Valid Values: 0 ohms to +inf ohms

Default Value: Search ni.com for Supported Properties by Device for the default value by instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_impedance_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_impedance_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Impedance Range:Impedance Range**
- C Attribute: **NIDCPOWER_ATTR_LCR_IMPEDANCE_RANGE**

lcr_impedance_range_source

nidcpower.Session.**lcr_impedance_range_source**

Specifies how the impedance range for LCR measurements is determined.

“nidcpower.Session.LCR_IMPEDANCE_AUTORANGE overrides any impedance range determined by this property.

“

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Note: One or more of the referenced properties are not in the Python API for this driver.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_impedance_range_source`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_impedance_range_source`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCRImpedanceRangeSource
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Impedance Range:Advanced:Impedance Range Source**
- C Attribute: **NIDCPOWER_ATTR_LCR_IMPEDANCE_RANGE_SOURCE**

---

lcr_load_capacitance

nidcpower.Session.**lcr_load_capacitance**

Specifies the load capacitance, in farads and assuming a series model, of the DUT in order to compute the impedance range when the *nidcpower.Session.lcr_impedance_range_source* property is set to *LOAD_CONFIGURATION*.

Valid values: (0 farads, +inf farads) 0 is a special value that signifies +inf farads.

Default Value: Search ni.com for Supported Properties by Device for the default value by instrument

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_load_capacitance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_load_capacitance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Impedance Range:Advanced:Load Capacitance**
- C Attribute: **NIDCPOWER_ATTR_LCR_LOAD_CAPACITANCE**

lcr_load_compensation_enabled

nidcpower.Session.**lcr_load_compensation_enabled**

Specifies whether to apply load LCR compensation data to LCR measurements. Both the *nidcpower.Session.lcr_open_compensation_enabled* and *nidcpower.Session.lcr_short_compensation_enabled* properties must be set to True in order to set this property to True.

Use the *nidcpower.Session.lcr_open_short_load_compensation_data_source* property to define where the load compensation data that is applied to LCR measurements comes from.

Note: Load compensation data are applied only for those specific frequencies you define with *nidcpower.Session.perform_lcr_load_compensation()*; load compensation is not interpolated from the specific frequencies you define and applied to other frequencies.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_load_compensation_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_load_compensation_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Load:Enabled**
- C Attribute: **NIDCPOWER_ATTR_LCR_LOAD_COMPENSATION_ENABLED**

lcr_load_inductance

`nidcpower.Session.lcr_load_inductance`

Specifies the load inductance, in henrys and assuming a series model, of the DUT in order to compute the impedance range when the *nidcpower.Session.lcr_impedance_range_source* property is set to *LOAD_CONFIGURATION*.

Valid values: [0 henrys, +inf henrys)

Default Value: Search ni.com for Supported Properties by Device for the default value by instrument

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_load_inductance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_load_inductance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Impedance Range:Advanced:Load Inductance**

- C Attribute: **NIDCPOWER_ATTR_LCR_LOAD_INDUCTANCE**

lcr_load_resistance

nidcpower.Session.**lcr_load_resistance**

Specifies the load resistance, in ohms and assuming a series model, of the DUT in order to compute the impedance range when the *nidcpower.Session.lcr_impedance_range_source* property is set to *LOAD_CONFIGURATION*.

Valid values: [0 ohms, +inf ohms)

Default Value: Search ni.com for Supported Properties by Device for the default value by instrument

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_load_resistance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_load_resistance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Impedance Range:Advanced:Load Resistance**
- C Attribute: **NIDCPOWER_ATTR_LCR_LOAD_RESISTANCE**

lcr_measured_load_reactance

nidcpower.Session.**lcr_measured_load_reactance**

Specifies the reactance, in ohms, of the load used for load LCR compensation as measured by the instrument. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_measured_load_reactance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_measured_load_reactance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Load:Measured Reactance**
- C Attribute: **NIDCPOWER_ATTR_LCR_MEASURED_LOAD_REACTANCE**

lcr_measured_load_resistance

nidcpower.Session.**lcr_measured_load_resistance**

Specifies the resistance, in ohms, of the load used for load LCR compensation as measured by the instrument. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_measured_load_resistance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_measured_load_resistance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Load:Measured Resistance**
- C Attribute: **NIDCPOWER_ATTR_LCR_MEASURED_LOAD_RESISTANCE**

lcr_measurement_time

nidcpower.Session.**lcr_measurement_time**

Selects a general aperture time profile for LCR measurements. The actual duration of each profile depends on the frequency of the LCR test signal.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_measurement_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_measurement_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCRMeasurementTime
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Measurement Time**
- C Attribute: **NIDCPOWER_ATTR_LCR_MEASUREMENT_TIME**

lcr_open_compensation_enabled

nidcpower.Session.**lcr_open_compensation_enabled**

Specifies whether to apply open LCR compensation data to LCR measurements. Use the *nidcpower.Session.lcr_open_short_load_compensation_data_source* property to define where the open compensation data that is applied to LCR measurements comes from.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_open_compensation_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_open_compensation_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Open:Enabled**
- C Attribute: **NIDCPOWER_ATTR_LCR_OPEN_COMPENSATION_ENABLED**

lcr_open_conductance

nidcpower.Session.**lcr_open_conductance**

Specifies the conductance, in siemens, of the circuit used for open LCR compensation. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_open_conductance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_open_conductance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Open:Conductance**
- C Attribute: **NIDCPOWER_ATTR_LCR_OPEN_CONDUCTANCE**

lcr_open_short_load_compensation_data_source

nidcpower.Session.**lcr_open_short_load_compensation_data_source**

Specifies the source of the LCR compensation data NI-DCPower applies to LCR measurements.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_open_short_load_compensation_data_source`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_open_short_load_compensation_data_source`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCROpenShortLoadCompensationDataSource
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:LCR Open/Short/Load Compensation Data Source**
- C Attribute: **NIDCPOWER_ATTR_LCR_OPEN_SHORT_LOAD_COMPENSATION_DATA_SOURCE**

lcr_open_susceptance

nidcpower.Session.**lcr_open_susceptance**

Specifies the susceptance, in siemens, of the circuit used for open LCR compensation. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_open_susceptance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_open_susceptance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Open:Susceptance**
- C Attribute: **NIDCPOWER_ATTR_LCR_OPEN_SUSCEPTANCE**

lcr_short_compensation_enabled

nidcpower.Session.**lcr_short_compensation_enabled**

Specifies whether to apply short LCR compensation data to LCR measurements. Use the *nidcpower.Session.lcr_open_short_load_compensation_data_source* property to define where the short compensation data that is applied to LCR measurements comes from.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_short_compensation_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_short_compensation_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Short:Enabled**
- C Attribute: **NIDCPOWER_ATTR_LCR_SHORT_COMPENSATION_ENABLED**

lcr_short_custom_cable_compensation_enabled

nidcpower.Session.**lcr_short_custom_cable_compensation_enabled**

Defines how to apply short custom cable compensation in LCR mode when *nidcpower.Session.cable_length* property is set to *CUSTOM_ONBOARD_STORAGE* or *CUSTOM_AS_CONFIGURED*.

LCR custom cable compensation uses compensation data for both an open and short configuration. For open custom cable compensation, you must supply your own data from a call to *nidcpower.Session.perform_lcr_open_custom_cable_compensation()*. For short custom cable compensation, you can supply your own data from a call to *nidcpower.Session.perform_lcr_short_custom_cable_compensation()* or NI-DCPower can apply a default set of short compensation data.

False	Uses default short compensation data.
True	Uses short custom cable compensation data generated by *nidcpower.Session.perform_lcr_short_custom_cable_compensation()*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_short_custom_cable_compensation_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_short_custom_cable_compensation_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:LCR Short Custom Cable Compensation Enabled**
- C Attribute: **NIDCPOWER_ATTR_LCR_SHORT_CUSTOM_CABLE_COMPENSATION_ENABLED**

lcr_short_reactance

nidcpower.Session.**lcr_short_reactance**

Specifies the reactance, in ohms, of the circuit used for short LCR compensation. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_short_reactance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_short_reactance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Short:Reactance**
- C Attribute: **NIDCPOWER_ATTR_LCR_SHORT_REACTANCE**

---

lcr_short_resistance

nidcpower.Session.**lcr_short_resistance**

Specifies the resistance, in ohms, of the circuit used for short LCR compensation. This property applies when *nidcpower.Session.lcr_open_short_load_compensation_data_source* is set to *AS_DEFINED*.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_short_resistance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_short_resistance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Compensation:Short:Resistance**
- C Attribute: **NIDCPOWER_ATTR_LCR_SHORT_RESISTANCE**

lcr_source_aperture_time

`nidcpower.Session.lcr_source_aperture_time`

Specifies the LCR source aperture time for a channel, in seconds.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_source_aperture_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_source_aperture_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Advanced:Source Aperture Time**
- C Attribute: **NIDCPOWER_ATTR_LCR_SOURCE_APERTURE_TIME**

lcr_source_delay_mode

nidcpower.Session.**lcr_source_delay_mode**

For instruments in LCR mode, determines whether NI-DCPower automatically calculates and applies the source delay or applies a source delay you set manually.

You can return the source delay duration for either option by reading *nidcpower.Session.source_delay*.

When you use this property to manually set the source delay, it is possible to set source delays short enough to unbalance the bridge and affect measurement accuracy. LCR measurement methods report whether the bridge is unbalanced.

Default Value: *AUTOMATIC*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_source_delay_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_source_delay_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCRSourceDelayMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:Source Delay Mode**
- C Attribute: **NIDCPOWER_ATTR_LCR_SOURCE_DELAY_MODE**

---

lcr_stimulus_function

nidcpower.Session.**lcr_stimulus_function**

Specifies the type of test signal to apply to the DUT for LCR measurements.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_stimulus_function`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_stimulus_function`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCRStimulusFunction
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Function**
- C Attribute: **NIDCPOWER_ATTR_LCR_STIMULUS_FUNCTION**

lcr_voltage_amplitude

`nidcpower.Session.`**`lcr_voltage_amplitude`**

Specifies the amplitude, in volts RMS, of the AC voltage test signal applied to the DUT for LCR measurements. This property applies when the *nidcpower.Session.lcr_stimulus_function* property is set to *VOLTAGE*.

Valid Values: 7.08e-4 V RMS to 7.07 V RMS

Instrument specifications affect the valid values you can program. Refer to the specifications for your instrument for more information.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_voltage_amplitude`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_voltage_amplitude`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Voltage Amplitude**
- C Attribute: **NIDCPOWER_ATTR_LCR_VOLTAGE_AMPLITUDE**

lcr_voltage_range

nidcpower.Session.**lcr_voltage_range**

Specifies the voltage range, in volts RMS, for the specified channel(s). The range defines the valid values to which you can set the *nidcpower.Session.lcr_voltage_amplitude*. For valid ranges, refer to the specifications for your instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].lcr_voltage_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.lcr_voltage_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **LCR:AC Stimulus:Advanced:Voltage Range**
- C Attribute: **NIDCPOWER_ATTR_LCR_VOLTAGE_RANGE**

logical_name

nidcpower.Session.**logical_name**

Contains the logical name you specified when opening the current IVI session. You can pass a logical name to the `nidcpower.Session.__init__()` method. The IVI Configuration utility must contain an entry for the logical name. The logical name entry refers to a method section in the IVI Configuration file. The method section specifies a physical device and initial user options.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Advanced Session Information:Logical Name**
- C Attribute: **NIDCPOWER_ATTR_LOGICAL_NAME**

measure_buffer_size

nidcpower.Session.**measure_buffer_size**

Specifies the number of samples that the active channel measurement buffer can hold. The default value is the maximum number of samples that a device is capable of recording in one second. Valid Values: The PXIe-4051, PXIe-4147, and PXIe-4151 support values from 170 to 18000110. The PXIe-4162/4163 supports values from 256 to 1000192. The PXIe-4190 supports values from 102 to 6000048. The PXIe-4112, PXIe-4113, and PXIe-4154 support values from 1000 to 178956970. All other supported instruments support values from 1000 to 268435455. Default Value: Varies by device. Refer to Supported Properties by Device topic in the NI DC Power Supplies and SMUs Help for more information about default values.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_buffer_size`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_buffer_size`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Measure Buffer Size**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_BUFFER_SIZE**

measure_complete_event_delay

nidcpower.Session.**measure_complete_event_delay**

Specifies the amount of time to delay the generation of the Measure Complete event, in seconds. Valid Values: The PXIe-4051 supports values from 0 seconds to 39 seconds. The PXIe-4147 supports values from 0 seconds to 26.5 seconds. The PXIe-4151 supports values from 0 seconds to 42 seconds. The PXIe-4162/4163 and PXIe-4190 support values from 0 seconds to 23 seconds. All other supported instruments support values from 0 to 167 seconds. Default Value: Varies by device. Refer to Supported Properties by Device topic in the NI DC Power Supplies and SMUs Help for more information about default values.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_complete_event_delay`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_complete_event_delay`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Measure Complete Event:Event Delay**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_COMPLETE_EVENT_DELAY**

measure_complete_event_output_behavior

nidcpower.Session.**measure_complete_event_output_behavior**

Determines the event type's behavior when a corresponding trigger is received. If you set the Measure Complete event output behavior to *PULSE*, a single pulse is transmitted. If you set the Measure Complete event output behavior to *TOGGLE*, the output level toggles between low and high. The default value is *PULSE*.

Note: This property is not supported by all output terminals. This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_complete_event_output_behavior`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_complete_event_output_behavior`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventOutputBehavior
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Measure Complete Event:Output Behavior**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_COMPLETE_EVENT_OUTPUT_BEHAVIOR**

measure_complete_event_output_terminal

nidcpower.Session.**measure_complete_event_output_terminal**

Specifies the output terminal for exporting the Measure Complete event. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_complete_event_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_complete_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Measure Complete Event:Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_COMPLETE_EVENT_OUTPUT_TERMINAL**

measure_complete_event_pulse_polarity

`nidcpower.Session.`**`measure_complete_event_pulse_polarity`**

Specifies the behavior of the Measure Complete event. Default Value: *HIGH*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_complete_event_pulse_polarity`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_complete_event_pulse_polarity`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Polarity
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Measure Complete Event:Pulse:Polarity**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_COMPLETE_EVENT_PULSE_POLARITY**

measure_complete_event_pulse_width

nidcpower.Session.**measure_complete_event_pulse_width**

Specifies the width of the Measure Complete event, in seconds. The minimum event pulse width value for PXI devices is 150 ns, and the minimum event pulse width value for PXI Express devices is 250 ns. The maximum event pulse width value for all devices is 1.6 microseconds. Valid Values: 1.5e-7 to 1.6e-6 Default Value: The default value for PXI devices is 150 ns. The default value for PXI Express devices is 250 ns.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_complete_event_pulse_width`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_complete_event_pulse_width`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Measure Complete Event:Pulse:Width**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_COMPLETE_EVENT_PULSE_WIDTH**

---

measure_complete_event_toggle_initial_state

nidcpower.Session.**measure_complete_event_toggle_initial_state**

Specifies the initial state of the Measure Complete event when you set the *nidcpower.Session.measure_complete_event_output_behavior* property to *TOGGLE*. For a Single Point mode acquisition, if you set the initial state to `NIDCPOWER_VAL_LOW_STATE`, the output is set to low at session commit. The output switches to high when the event occurs during the acquisition. If you set the initial state to `NIDCPOWER_VAL_HIGH_STATE`, the output is set to a high state at session commit. The output switches to low when the event occurs during the acquisition. For a Sequence mode operation, if you set the initial state to `NIDCPOWER_VAL_LOW_STATE`, the output is set to low at session commit. The output switches to high the first time an event occurs during the acquisition. The second time an event occurs, the output switches to low. This pattern repeats for any subsequent event occurrences. If you set the initial state to `NIDCPOWER_VAL_HIGH_STATE`, the output is set to high at session commit. The output switches to low on the first time the event occurs during the acquisition. The second

time the event occurs, the output switches to high. This pattern repeats for any subsequent event occurrences. The default value is NIDCPOWER_VAL_LOW_STATE.

Note: This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com

Note: One or more of the referenced values are not in the Python API for this driver. Enums that only define values, or represent True/False, have been removed.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_complete_event_toggle_initial_state`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_complete_event_toggle_initial_state`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventToggleInitialState
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Measure Complete Event:Toggle:Initial State**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_COMPLETE_EVENT_TOGGLE_INITIAL_STATE**

measure_record_delta_time

nidcpower.Session.**measure_record_delta_time**

Queries the amount of time, in seconds, between between the start of two consecutive measurements in a measure record. Only query this property after the desired measurement settings are committed. two measurements and the rest would differ.

Note: This property is not available when Auto Zero is configured to Once because the amount of time between the first

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_record_delta_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_record_delta_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read only
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Measure Record Delta Time**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_RECORD_DELTA_TIME**

measure_record_length

`nidcpower.Session.`**`measure_record_length`**

Specifies how many measurements compose a measure record. When this property is set to a value greater than 1, the *nidcpower.Session.measure_when* property must be set to *AUTOMATICALLY_AFTER_SOURCE_COMPLETE* or *ON_MEASURE_TRIGGER*. Valid Values: 1 to 16,777,216 Default Value: 1

Note: This property is not available in a session involving multiple channels.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_record_length`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_record_length`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Measure Record Length**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_RECORD_LENGTH**

measure_record_length_is_finite

nidcpower.Session.**measure_record_length_is_finite**

Specifies whether to take continuous measurements. Call the *nidcpower.Session.abort()* method to stop continuous measurements. When this property is set to False and the *nidcpower.Session.source_mode* property is set to *SINGLE_POINT*, the *nidcpower.Session.measure_when* property must be set to *AUTOMATICALLY_AFTER_SOURCE_COMPLETE* or *ON_MEASURE_TRIGGER*. When this property is set to False and the *nidcpower.Session.source_mode* property is set to *SEQUENCE*, the *nidcpower.Session.measure_when* property must be set to *ON_MEASURE_TRIGGER*. Default Value: True

---

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device. This property is not available in a session involving multiple channels.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_record_length_is_finite`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_record_length_is_finite`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Measure Record Length Is Finite**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_RECORD_LENGTH_IS_FINITE**

---

measure_trigger_type

nidcpower.Session.**measure_trigger_type**

Specifies the behavior of the Measure trigger. Default Value: *DIGITAL_EDGE*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_trigger_type`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_trigger_type`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TriggerType
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Measure Trigger:Trigger Type**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_TRIGGER_TYPE**

measure_when

nidcpower.Session.**measure_when**

Specifies when the measure unit should acquire measurements. Unless this property is configured to *ON_MEASURE_TRIGGER*, the *nidcpower.Session.measure_trigger_type* property is ignored. Refer to the Acquiring Measurements topic in the NI DC Power Supplies and SMUs Help for more information about how to configure your measurements. Default Value: If the *nidcpower.Session.source_mode* property is set to *SINGLE_POINT*, the default value is *ON_DEMAND*. This value supports only the *nidcpower.Session.measure()* method and *nidcpower.Session.measure_multiple()* method. If the *nidcpower.Session.source_mode* property is set to *SEQUENCE*, the default value is *AUTOMATICALLY_AFTER_SOURCE_COMPLETE*. This value supports only the *nidcpower.Session.fetch_multiple()* method.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].measure_when`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.measure_when`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.MeasureWhen
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Measure When**
- C Attribute: **NIDCPOWER_ATTR_MEASURE_WHEN**

merged_channels

nidcpower.Session.**merged_channels**

Specifies the channel(s) to merge with a designated primary channel of an instrument in order to increase the maximum current you can source from the instrument. This property designates the merge channels to combine with a primary channel. To designate the primary channel, initialize the session to the primary channel only. Note: You cannot change the merge configuration with this property when the session is in the Running state. For complete information on using merged channels with this property, refer to Merged Channels in the NI DC Power Supplies and SMUs Help.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device. Devices that do not support this property behave as if no channels were merged. Default Value: Refer to the Supported Properties by Device topic for the default value by device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].merged_channels`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.merged_channels`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Merged Channels**
- C Attribute: **NIDCPOWER_ATTR_MERGED_CHANNELS**

output_capacitance

nidcpower.Session.**output_capacitance**

Specifies whether to use a low or high capacitance on the output for the specified channel(s). Refer to the NI PXI-4130 Output Capacitance Selection topic in the NI DC Power Supplies and SMUs Help for more information about capacitance.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_capacitance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_capacitance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.OutputCapacitance
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Output Capacitance**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CAPACITANCE**

output_connected

nidcpower.Session.**output_connected**

Specifies whether the output relay is connected (closed) or disconnected (open). The *nidcpower.Session.output_enabled* property does not change based on this property; they are independent of each other.

Set this property to False to disconnect the output terminal from the output.

Default Value: True

Note: Only disconnect the output when disconnecting is necessary for your application. For example, a battery connected to the output terminal might discharge unless the relay is disconnected. Excessive connecting and disconnecting of the output can cause premature wear on electromechanical relays, such as those used by the PXIe-4147, PXI-4132, or PXIe-4138/39.

The PXIe-4051 does not have an output relay. For the PXIe-4051, this property specifies whether the input MOSFETs are connected (ON) or disconnected (OFF).

NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_connected`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_connected`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Connected**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CONNECTED**

output_cutoff_current_change_limit_high

`nidcpower.Session.`**`output_cutoff_current_change_limit_high`**

Specifies a limit for positive current slew rate, in amps per microsecond, for output cutoff. If the current increases at a rate that exceeds this limit, the output is disconnected.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *CURRENT_CHANGE_HIGH* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_current_change_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_current_change_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Current Change Limit High**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_CURRENT_CHANGE_LIMIT_HIGH**

output_cutoff_current_change_limit_low

nidcpower.Session.**output_cutoff_current_change_limit_low**

Specifies a limit for negative current slew rate, in amps per microsecond, for output cutoff. If the current decreases at a rate that exceeds this limit, the output is disconnected.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *CURRENT_CHANGE_LOW* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_current_change_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_current_change_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Current Change Limit Low**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_CURRENT_CHANGE_LIMIT_LOW**

output_cutoff_current_measure_limit_high

nidcpower.Session.**output_cutoff_current_measure_limit_high**

Specifies a high limit current value, in amps, for output cutoff. If the measured current exceeds this limit, the output is disconnected.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *CURRENT_MEASURE_HIGH* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_current_measure_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_current_measure_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Current Measure Limit High**

- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_CURRENT_MEASURE_LIMIT_HIGH**

output_cutoff_current_measure_limit_low

nidcpower.Session.**output_cutoff_current_measure_limit_low**

Specifies a low limit current value, in amps, for output cutoff. If the measured current falls below this limit, the output is disconnected.

To find out whether an output has fallen below this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *CURRENT_MEASURE_LOW* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_current_measure_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_current_measure_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Current Measure Limit Low**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_CURRENT_MEASURE_LIMIT_LOW**

output_cutoff_current_overrange_enabled

nidcpower.Session.**output_cutoff_current_overrange_enabled**

Enables or disables current overrange functionality for output cutoff. If enabled, the output is disconnected when the measured current saturates the current range.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *CURRENT_SATURATED* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_current_overrange_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_current_overrange_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Current Overrange Enabled**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_CURRENT_OVERRANGE_ENABLED**

---

output_cutoff_delay

nidcpower.Session.**output_cutoff_delay**

Delays disconnecting the output by the time you specify, in seconds, when a limit is exceeded.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_delay`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_delay`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Delay**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_DELAY**

output_cutoff_enabled

`nidcpower.Session.`**`output_cutoff_enabled`**

Enables or disables output cutoff functionality. If enabled, you can define output cutoffs that, if exceeded, cause the output of the specified channel(s) to be disconnected. When this property is disabled, all other output cutoff properties are ignored.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Instruments that do not support this property behave as if this property were set to False.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Enabled**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_ENABLED**

output_cutoff_voltage_change_limit_high

nidcpower.Session.**output_cutoff_voltage_change_limit_high**

Specifies a limit for positive voltage slew rate, in volts per microsecond, for output cutoff. If the voltage increases at a rate that exceeds this limit, the output is disconnected.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* with *VOLTAGE_CHANGE_HIGH* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_voltage_change_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_voltage_change_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Voltage Change Limit High**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_VOLTAGE_CHANGE_LIMIT_HIGH**

output_cutoff_voltage_change_limit_low

nidcpower.Session.**output_cutoff_voltage_change_limit_low**

Specifies a limit for negative voltage slew rate, in volts per microsecond, for output cutoff. If the voltage decreases at a rate that exceeds this limit, the output is disconnected.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* with *VOLTAGE_CHANGE_LOW* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_voltage_change_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_voltage_change_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Voltage Change Limit Low**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_VOLTAGE_CHANGE_LIMIT_LOW**

---

output_cutoff_voltage_measure_limit_high

nidcpower.Session.**output_cutoff_voltage_measure_limit_high**

Specifies a high limit voltage value, in volts, for output cutoff. If the measured voltage exceeds this limit, the output is disconnected.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *VOLTAGE_MEASURE_HIGH* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_voltage_measure_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_voltage_measure_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Voltage Measure Limit High**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_VOLTAGE_MEASURE_LIMIT_HIGH**

output_cutoff_voltage_measure_limit_low

`nidcpower.Session.`**`output_cutoff_voltage_measure_limit_low`**

Specifies a low limit voltage value, in volts, for output cutoff. If the measured voltage falls below this limit, the output is disconnected.

To find out whether an output has fallen below this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *VOLTAGE_MEASURE_LOW* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_voltage_measure_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_voltage_measure_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Voltage Measure Limit Low**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_VOLTAGE_MEASURE_LIMIT_LOW**

output_cutoff_voltage_output_limit_high

nidcpower.Session.**output_cutoff_voltage_output_limit_high**

Specifies a high limit voltage value, in volts, for output cutoff. If the voltage output exceeds this limit, the output is disconnected.

To find out whether an output has exceeded this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *VOLTAGE_OUTPUT_HIGH* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_voltage_output_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_voltage_output_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Voltage Output Limit High**

- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_VOLTAGE_OUTPUT_LIMIT_HIGH**

output_cutoff_voltage_output_limit_low

nidcpower.Session.**output_cutoff_voltage_output_limit_low**

Specifies a low limit voltage value, in volts, for output cutoff. If the voltage output falls below this limit, the output is disconnected.

To find out whether an output has fallen below this limit, call the *nidcpower.Session.query_latched_output_cutoff_state()* method with *VOLTAGE_OUTPUT_LOW* as the output cutoff reason.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_cutoff_voltage_output_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_cutoff_voltage_output_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Cutoff:Voltage Output Limit Low**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_CUTOFF_VOLTAGE_OUTPUT_LIMIT_LOW**

output_enabled

nidcpower.Session.**output_enabled**

Specifies whether the output is enabled (True) or disabled (False). Depending on the value you specify for the *nidcpower.Session.output_function* property, you also must set the voltage level or current level in addition to enabling the output

Default Value: The default value is True if you use the `nidcpower.Session.__init__()` method to open the session. Otherwise the default value is False, including when you use a calibration session or the deprecated programming model.

Note: If the session is in the Committed or Uncommitted states, enabling the output does not take effect until you call the *nidcpower.Session.initiate()* method. Refer to the Programming States topic in the NI DC Power Supplies and SMUs Help for more information about NI-DCPower programming states.

NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Enabled**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_ENABLED**

---

output_function

nidcpower.Session.**output_function**

Configures the method to generate on the specified channel(s). When *DC_VOLTAGE* is selected, the device generates the desired voltage level on the output as long as the output current is below the current limit. You can use the following properties to configure the channel when *DC_VOLTAGE* is selected: *nidcpower.Session.voltage_level* *nidcpower.Session.current_limit* *nidcpower.Session.current_limit_high* *nidcpower.Session.current_limit_low* *nidcpower.Session.voltage_level_range* *nidcpower.Session.current_limit_range* *nidcpower.Session.compliance_limit_symmetry* When *DC_CURRENT* is selected, the device generates the desired current level on the output as long as the output voltage is below the voltage limit. You can use the following properties to configure the channel when *DC_CURRENT* is selected: *nidcpower.Session.current_level* *nidcpower.Session.voltage_limit* *nidcpower.Session.voltage_limit_high* *nidcpower.Session.voltage_limit_low* *nidcpower.Session.current_level_range* *nidcpower.Session.voltage_limit_range* *nidcpower.Session.compliance_limit_symmetry*

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_function`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_function`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.OutputFunction
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Function**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_FUNCTION**

---

output_resistance

nidcpower.Session.**output_resistance**

Specifies the output resistance that the device attempts to generate for the specified channel(s). This property is available only when you set the *nidcpower.Session.output_function* property on a support device. Refer to a supported device's topic about output resistance for more information about selecting an output resistance. about supported devices. Default Value: 0.0

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].output_resistance`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.output_resistance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Output Resistance**
- C Attribute: **NIDCPOWER_ATTR_OUTPUT_RESISTANCE**

---

overranging_enabled

nidcpower.Session.**overranging_enabled**

Specifies whether NI-DCPower allows setting the voltage level, current level, voltage limit and current limit outside the device specification limits. True means that overranging is enabled. Refer to the Ranges topic in the NI DC Power Supplies and SMUs Help for more information about overranging. Default Value: False

---

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].overranging_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.overranging_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Overranging Enabled**
- C Attribute: **NIDCPOWER_ATTR_OVERRANGING_ENABLED**

ovp_enabled

`nidcpower.Session.`**`ovp_enabled`**

Enables (True) or disables (False) overvoltage protection (OVP). Refer to the Output Overvoltage Protection topic in the NI DC Power Supplies and SMUs Help for more information about overvoltage protection. Default Value: False

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].ovp_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.ovp_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:OVP Enabled**
- C Attribute: **NIDCPOWER_ATTR_OVP_ENABLED**

ovp_limit

nidcpower.Session.**ovp_limit**

Determines the voltage limit, in volts, beyond which overvoltage protection (OVP) engages. The limit is specified as a positive value, but symmetric positive and negative limits are enforced simultaneously. For example, setting the OVP Limit to 65 will configure the OVP feature to trigger an OVP error if the output exceeds ±65 V.

Valid Values: 2 V to 210 V Default Value: 210 V

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].ovp_limit`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.ovp_limit`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:OVP Limit**
- C Attribute: **NIDCPOWER_ATTR_OVP_LIMIT**

---

power_allocation_mode

nidcpower.Session.**power_allocation_mode**

Determines whether the device sources the power its source configuration requires or a specific wattage you request; determines whether NI-DCPower proactively checks that this sourcing power is within the maximum per-channel and overall sourcing power of the device.

When this property configures NI-DCPower to perform a sourcing power check, a device is not permitted to source power in excess of its maximum per-channel or overall sourcing power. If the check determines a source configuration or power request would require the device to do so, NI-DCPower returns an error.

When this property does not configure NI-DCPower to perform a sourcing power check, a device can attempt to fulfill source configurations that would require it to source power in excess of its maximum per-channel or overall sourcing power and may shut down to prevent damage.

Default Value: Refer to the Supported Properties by Device topic for the default value by device.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device. Devices that do not support this property behave as if this property were set to *DISABLED*.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].power_allocation_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.power_allocation_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.PowerAllocationMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Power Allocation Mode**
- C Attribute: **NIDCPOWER_ATTR_POWER_ALLOCATION_MODE**

---

power_line_frequency

`nidcpower.Session.`**`power_line_frequency`**

Specifies the power line frequency for specified channel(s). NI-DCPower uses this value to select a timebase for setting the *nidcpower.Session.aperture_time* property in power line cycles (PLCs). in the NI DC Power Supplies and SMUs Help for information about supported devices. Default Value: `NIDCPOWER_VAL_60_HERTZ`

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Note: One or more of the referenced values are not in the Python API for this driver. Enums that only define values, or represent True/False, have been removed.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].power_line_frequency`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.power_line_frequency`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Power Line Frequency**
- C Attribute: **NIDCPOWER_ATTR_POWER_LINE_FREQUENCY**

power_source

`nidcpower.Session.power_source`

Specifies the power source to use. NI-DCPower switches the power source used by the device to the specified value. Default Value: *AUTOMATIC* is set to *AUTOMATIC*. However, if the session is in the Committed or Uncommitted state when you set this property, the power source selection only occurs after you call the *nidcpower.Session.initiate()* method.

Note: Automatic selection is not persistent and occurs only at the time this property

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.PowerSource
Permissions	read-write
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Advanced:Power Source**

- C Attribute: **NIDCPOWER_ATTR_POWER_SOURCE**

power_source_in_use

nidcpower.Session.**power_source_in_use**

Indicates whether the device is using the internal or auxiliary power source to generate power.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.PowerSourceInUse
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Advanced:Power Source In Use**
- C Attribute: **NIDCPOWER_ATTR_POWER_SOURCE_IN_USE**

pulse_bias_current_level

nidcpower.Session.**pulse_bias_current_level**

Specifies the pulse bias current level, in amps, that the device attempts to generate on the specified channel(s) during the off phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_current_level_range* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_current_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_current_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Bias Current Level**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_CURRENT_LEVEL**

pulse_bias_current_limit

nidcpower.Session.**pulse_bias_current_limit**

Specifies the pulse bias current limit, in amps, that the output cannot exceed when generating the desired pulse bias voltage on the specified channel(s) during the off phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_current_limit_range* property.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_current_limit`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_current_limit`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Bias Current Limit**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_CURRENT_LIMIT**

pulse_bias_current_limit_high

nidcpower.Session.**pulse_bias_current_limit_high**

Specifies the maximum current, in amps, that the output can produce when generating the desired pulse voltage on the specified channel(s) during the *off* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. You must also specify a *nidcpower.Session.pulse_bias_current_limit_low* to complete the asymmetric range. **Valid Values:** [1% of *nidcpower.Session.pulse_current_limit_range*, *nidcpower.Session.pulse_current_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

---

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_current_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_current_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Bias Current Limit High**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_CURRENT_LIMIT_HIGH**

---

pulse_bias_current_limit_low

nidcpower.Session.**pulse_bias_current_limit_low**

Specifies the minimum current, in amps, that the output can produce when generating the desired pulse voltage on the specified channel(s) during the *off* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. You must also specify a *nidcpower.Session.pulse_bias_current_limit_high* to complete the asymmetric range. **Valid Values:** [-*nidcpower.Session.pulse_current_limit_range*, -1% of *nidcpower.Session.pulse_current_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

---

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_current_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_current_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Bias Current Limit Low**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_CURRENT_LIMIT_LOW**

---

pulse_bias_delay

nidcpower.Session.**pulse_bias_delay**

Determines when, in seconds, the device generates the Pulse Complete event after generating the off level of a pulse. Valid Values: 0 to 167 seconds Default Value: 16.67 milliseconds

---

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_delay`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_delay`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Pulse Bias Delay**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_DELAY**

---

pulse_bias_voltage_level

nidcpower.Session.**pulse_bias_voltage_level**

Specifies the pulse bias voltage level, in volts, that the device attempts to generate on the specified channel(s) during the off phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_voltage_level_range* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_voltage_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_voltage_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Bias Voltage Level**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_VOLTAGE_LEVEL**

pulse_bias_voltage_limit

`nidcpower.Session.`**`pulse_bias_voltage_limit`**

Specifies the pulse voltage limit, in volts, that the output cannot exceed when generating the desired current on the specified channel(s) during the off phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_voltage_limit_range* property.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_voltage_limit`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_voltage_limit`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Bias Voltage Limit**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_VOLTAGE_LIMIT**

pulse_bias_voltage_limit_high

nidcpower.Session.**pulse_bias_voltage_limit_high**

Specifies the maximum voltage, in volts, that the output can produce when generating the desired pulse current on the specified channel(s) during the *off* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. You must also specify a *nidcpower.Session.pulse_bias_voltage_limit_low* to complete the asymmetric range. **Valid Values:** [1% of *nidcpower.Session.pulse_voltage_limit_range*, *nidcpower.Session.pulse_voltage_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_voltage_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_voltage_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Bias Voltage Limit High**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_VOLTAGE_LIMIT_HIGH**

pulse_bias_voltage_limit_low

nidcpower.Session.**pulse_bias_voltage_limit_low**

Specifies the minimum voltage, in volts, that the output can produce when generating the desired pulse current on the specified channel(s) during the *off* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. You must also specify a *nidcpower.Session.pulse_bias_voltage_limit_high* to complete the asymmetric range. **Valid Values:** [-*nidcpower.Session.pulse_voltage_limit_range*, -1% of *nidcpower.Session.pulse_voltage_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

---

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_bias_voltage_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_bias_voltage_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Bias Voltage Limit Low**
- C Attribute: **NIDCPOWER_ATTR_PULSE_BIAS_VOLTAGE_LIMIT_LOW**

---

pulse_complete_event_output_terminal

nidcpower.Session.**pulse_complete_event_output_terminal**

Specifies the output terminal for exporting the Pulse Complete event. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0. Default Value:The default value for PXI Express devices is 250 ns.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_complete_event_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_complete_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Pulse Complete Event:Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_PULSE_COMPLETE_EVENT_OUTPUT_TERMINAL**

---

pulse_complete_event_pulse_polarity

nidcpower.Session.**pulse_complete_event_pulse_polarity**

Specifies the behavior of the Pulse Complete event. Default Value: *HIGH*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_complete_event_pulse_polarity`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_complete_event_pulse_polarity`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Polarity
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Pulse Complete Event:Pulse:Polarity**
- C Attribute: **NIDCPOWER_ATTR_PULSE_COMPLETE_EVENT_PULSE_POLARITY**

pulse_complete_event_pulse_width

`nidcpower.Session.`**`pulse_complete_event_pulse_width`**

Specifies the width of the Pulse Complete event, in seconds. The minimum event pulse width value for PXI Express devices is 250 ns. The maximum event pulse width value for PXI Express devices is 1.6 microseconds. Default Value: The default value for PXI Express devices is 250 ns.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_complete_event_pulse_width`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_complete_event_pulse_width`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Pulse Complete Event:Pulse:Width**
- C Attribute: **NIDCPOWER_ATTR_PULSE_COMPLETE_EVENT_PULSE_WIDTH**

pulse_current_level

nidcpower.Session.**pulse_current_level**

Specifies the pulse current level, in amps, that the device attempts to generate on the specified channel(s) during the on phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_current_level_range* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_current_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_current_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Current Level**
- C Attribute: **NIDCPOWER_ATTR_PULSE_CURRENT_LEVEL**

---

pulse_current_level_range

nidcpower.Session.**pulse_current_level_range**

Specifies the pulse current level range, in amps, for the specified channel(s). The range defines the valid values to which you can set the pulse current level and pulse bias current level. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. For valid ranges, refer to the specifications for your instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_current_level_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_current_level_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Current Level Range**
- C Attribute: **NIDCPOWER_ATTR_PULSE_CURRENT_LEVEL_RANGE**

pulse_current_limit

nidcpower.Session.**pulse_current_limit**

Specifies the pulse current limit, in amps, that the output cannot exceed when generating the desired pulse voltage on the specified channel(s) during the on phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE* and the *nidcpower.Session.compliance_limit_symmetry* property is set to *SYMMETRIC*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_current_limit_range* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_current_limit`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_current_limit`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Current Limit**
- C Attribute: **NIDCPOWER_ATTR_PULSE_CURRENT_LIMIT**

pulse_current_limit_high

nidcpower.Session.**pulse_current_limit_high**

Specifies the maximum current, in amps, that the output can produce when generating the desired pulse voltage on the specified channel(s) during the *on* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. You must also specify a *nidcpower.Session.pulse_current_limit_low* to complete the asymmetric range. **Valid Values:** [1% of *nidcpower.Session.pulse_current_limit_range*, *nidcpower.Session.pulse_current_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_current_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_current_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Current Limit High**
- C Attribute: **NIDCPOWER_ATTR_PULSE_CURRENT_LIMIT_HIGH**

pulse_current_limit_low

nidcpower.Session.**pulse_current_limit_low**

Specifies the minimum current, in amps, that the output can produce when generating the desired pulse voltage on the specified channel(s) during the *on* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. You must also specify a *nidcpower.Session.pulse_current_limit_high* to complete the asymmetric range. **Valid Values:** [-*nidcpower.Session.pulse_current_limit_range*, -1% of *nidcpower.Session.pulse_current_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

---

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_current_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_current_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Current Limit Low**
- C Attribute: **NIDCPOWER_ATTR_PULSE_CURRENT_LIMIT_LOW**

---

pulse_current_limit_range

nidcpower.Session.**pulse_current_limit_range**

Specifies the pulse current limit range, in amps, for the specified channel(s). The range defines the valid values to which you can set the pulse current limit and pulse bias current limit. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. For valid ranges, refer to the specifications for your instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_current_limit_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_current_limit_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Current Limit Range**
- C Attribute: **NIDCPOWER_ATTR_PULSE_CURRENT_LIMIT_RANGE**

pulse_off_time

`nidcpower.Session.`**`pulse_off_time`**

Determines the length, in seconds, of the off phase of a pulse. Valid Values: 10 microseconds to 167 seconds Default Value: 34 milliseconds

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_off_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_off_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Pulse Off Time**
- C Attribute: **NIDCPOWER_ATTR_PULSE_OFF_TIME**

pulse_on_time

nidcpower.Session.**pulse_on_time**

Determines the length, in seconds, of the on phase of a pulse. Valid Values: 10 microseconds to 167 seconds Default Value: 34 milliseconds

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_on_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_on_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Pulse On Time**
- C Attribute: **NIDCPOWER_ATTR_PULSE_ON_TIME**

pulse_trigger_type

nidcpower.Session.**pulse_trigger_type**

Specifies the behavior of the Pulse trigger. Default Value: *NONE*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_trigger_type`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_trigger_type`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TriggerType
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Pulse Trigger:Trigger Type**
- C Attribute: **NIDCPOWER_ATTR_PULSE_TRIGGER_TYPE**

---

pulse_voltage_level

nidcpower.Session.**pulse_voltage_level**

Specifies the pulse current limit, in amps, that the output cannot exceed when generating the desired pulse voltage on the specified channel(s) during the on phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_VOLTAGE*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_current_limit_range* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_voltage_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_voltage_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Voltage Level**
- C Attribute: **NIDCPOWER_ATTR_PULSE_VOLTAGE_LEVEL**

pulse_voltage_level_range

`nidcpower.Session.`**`pulse_voltage_level_range`**

Specifies the pulse voltage level range, in volts, for the specified channel(s). The range defines the valid values at which you can set the pulse voltage level and pulse bias voltage level. This property is applicable only if the *`nidcpower.Session.output_function`* property is set to *`PULSE_VOLTAGE`*. For valid ranges, refer to the specifications for your instrument.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *`nidcpower.Session`* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_voltage_level_range`

To set/get on all channels, you can call the property directly on the *`nidcpower.Session`*.

Example: `my_session.pulse_voltage_level_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Voltage:Pulse Voltage Level Range**
- C Attribute: **NIDCPOWER_ATTR_PULSE_VOLTAGE_LEVEL_RANGE**

pulse_voltage_limit

nidcpower.Session.**pulse_voltage_limit**

Specifies the pulse voltage limit, in volts, that the output cannot exceed when generating the desired pulse current on the specified channel(s) during the on phase of a pulse. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT* and the *nidcpower.Session.compliance_limit_symmetry* property is set to *SYMMETRIC*. Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.pulse_voltage_limit_range* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_voltage_limit`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_voltage_limit`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Voltage Limit**
- C Attribute: **NIDCPOWER_ATTR_PULSE_VOLTAGE_LIMIT**

---

pulse_voltage_limit_high

nidcpower.Session.**pulse_voltage_limit_high**

Specifies the maximum voltage, in volts, that the output can produce when generating the desired pulse current on the specified channel(s) during the *on* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. You must also specify a *nidcpower.Session.pulse_voltage_limit_low* to complete the asymmetric range. **Valid Values:** [1% of *nidcpower.Session.pulse_voltage_limit_range*, *nidcpower.Session.pulse_voltage_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_voltage_limit_high`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_voltage_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Voltage Limit High**
- C Attribute: **NIDCPOWER_ATTR_PULSE_VOLTAGE_LIMIT_HIGH**

pulse_voltage_limit_low

`nidcpower.Session.`**`pulse_voltage_limit_low`**

Specifies the minimum voltage, in volts, that the output can produce when generating the desired pulse current on the specified channel(s) during the *on* phase of a pulse. This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. You must also specify a *nidcpower.Session.pulse_voltage_limit_high* to complete the asymmetric range. **Valid Values:** [-*nidcpower.Session.pulse_voltage_limit_range*, -1% of *nidcpower.Session.pulse_voltage_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True or if the *nidcpower.Session.output_function* property is set to a pulsing method.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_voltage_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_voltage_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Voltage Limit Low**
- C Attribute: **NIDCPOWER_ATTR_PULSE_VOLTAGE_LIMIT_LOW**

pulse_voltage_limit_range

`nidcpower.Session.`**`pulse_voltage_limit_range`**

Specifies the pulse voltage limit range, in volts, for the specified channel(s). The range defines the valid values to which you can set the pulse voltage limit and pulse bias voltage limit. This property is applicable only if the *nidcpower.Session.output_function* property is set to *PULSE_CURRENT*. For valid ranges, refer to the specifications for your instrument.

Note: The channel must be enabled for the specified current limit to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].pulse_voltage_limit_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.pulse_voltage_limit_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Pulse Current:Pulse Voltage Limit Range**
- C Attribute: **NIDCPOWER_ATTR_PULSE_VOLTAGE_LIMIT_RANGE**

query_instrument_status

nidcpower.Session.**query_instrument_status**

Specifies whether NI-DCPower queries the device status after each operation. Querying the device status is useful for debugging. After you validate your program, you can set this property to False to disable status checking and maximize performance. NI-DCPower ignores status checking for particular properties regardless of the setting of this property. Use the `nidcpower.Session.__init__()` method to override this value. Default Value: True

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:User Options:Query Instrument Status**
- C Attribute: **NIDCPOWER_ATTR_QUERY_INSTRUMENT_STATUS**

ready_for_pulse_trigger_event_output_terminal

nidcpower.Session.**ready_for_pulse_trigger_event_output_terminal**

Specifies the output terminal for exporting the Ready For Pulse Trigger event. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].ready_for_pulse_trigger_event_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.ready_for_pulse_trigger_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Ready For Pulse Trigger Event:Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_READY_FOR_PULSE_TRIGGER_EVENT_OUTPUT_TERMINAL**

ready_for_pulse_trigger_event_pulse_polarity

nidcpower.Session.**ready_for_pulse_trigger_event_pulse_polarity**

Specifies the behavior of the Ready For Pulse Trigger event. Default Value: *HIGH*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].ready_for_pulse_trigger_event_pulse_polarity`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.ready_for_pulse_trigger_event_pulse_polarity`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Polarity
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Ready For Pulse Trigger Event:Pulse:Polarity**
- C Attribute: **NIDCPOWER_ATTR_READY_FOR_PULSE_TRIGGER_EVENT_PULSE_POLARITY**

ready_for_pulse_trigger_event_pulse_width

nidcpower.Session.**ready_for_pulse_trigger_event_pulse_width**

Specifies the width of the Ready For Pulse Trigger event, in seconds. The minimum event pulse width value for PXI Express devices is 250 ns. The maximum event pulse width value for all devices is 1.6 microseconds. Default Value: The default value for PXI Express devices is 250 ns

---

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].ready_for_pulse_trigger_event_pulse_width`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.ready_for_pulse_trigger_event_pulse_width`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Ready For Pulse Trigger Event:Pulse:Width**
- C Attribute: **NIDCPOWER_ATTR_READY_FOR_PULSE_TRIGGER_EVENT_PULSE_WIDTH**

---

requested_power_allocation

nidcpower.Session.**requested_power_allocation**

Specifies the power, in watts, to request the device to source from each active channel.

This property defines the power to source from the device only if the *nidcpower.Session.power_allocation_mode* property is set to *MANUAL*.

The power you request with this property may be incompatible with the power a given source configuration requires or the power the device can provide: If the requested power is less than the power required for the source configuration, the device does not exceed the requested power, and NI-DCPower returns an error. If the requested power is greater than the maximum per-channel or overall sourcing power, the device does not exceed the allowed power, and NI-DCPower returns an error.

Valid Values: [0, device per-channel maximum power]

Default Value: Refer to the Supported Properties by Device topic for the default value by device.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].requested_power_allocation`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.requested_power_allocation`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Requested Power Allocation**
- C Attribute: **NIDCPOWER_ATTR_REQUESTED_POWER_ALLOCATION**

reset_average_before_measurement

nidcpower.Session.**reset_average_before_measurement**

Specifies whether the measurement returned from any measurement call starts with a new measurement call (True) or returns a measurement that has already begun or completed(False). When you set the *nidcpower.Session.samples_to_average* property in the Running state, the channel measurements might move out of synchronization. While NI-DCPower automatically synchronizes measurements upon the initialization of a session, you can force a synchronization in the running state before you run the *nidcpower.Session.measure_multiple()* method. To force a synchronization in the running state, set this property to True, and then run the *nidcpower.Session.measure_multiple()* method, specifying all channels in the channel name parameter. You can set the *nidcpower.Session.reset_average_before_measurement* property to False after the *nidcpower.Session.measure_multiple()* method completes. Default Value: True

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].reset_average_before_measurement`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.reset_average_before_measurement`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Advanced:Reset Average Before Measurement**
- C Attribute: **NIDCPOWER_ATTR_RESET_AVERAGE_BEFORE_MEASUREMENT**

samples_to_average

`nidcpower.Session.`**`samples_to_average`**

Specifies the number of samples to average when you take a measurement. Increasing the number of samples to average decreases measurement noise but increases the time required to take a measurement. Refer to the NI PXI-4110, NI PXI-4130, NI PXI-4132, or NI PXIe-4154 Averaging topic for optional property settings to improve immunity to certain noise types, or refer to the NI PXIe-4140/4141 DC Noise Rejection, NI PXIe-4142/4143 DC Noise Rejection, or NI PXIe-4144/4145 DC Noise Rejection topic for information about improving noise immunity for those devices. Default Value: NI PXI-4110 or NI PXI-4130—10 NI PXI-4132—1 NI PXIe-4112—1 NI PXIe-4113—1 NI PXIe-4140/4141—1 NI PXIe-4142/4143—1 NI PXIe-4144/4145—1 NI PXIe-4154—500

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].samples_to_average`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.samples_to_average`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Samples To Average**
- C Attribute: **NIDCPOWER_ATTR_SAMPLES_TO_AVERAGE**

self_calibration_persistence

nidcpower.Session.**self_calibration_persistence**

Specifies whether the values calculated during self-calibration should be written to hardware to be used until the next self-calibration or only used until the *nidcpower.Session.reset_device()* method is called or the machine is powered down. This property affects the behavior of the *nidcpower.Session.self_cal()* method. When set to *KEEP_IN_MEMORY*, the values calculated by the *nidcpower.Session.self_cal()* method are used in the existing session, as well as in all further sessions until you call the *nidcpower.Session.reset_device()* method or restart the machine. When you set this property to *WRITE_TO_EEPROM*, the values calculated by the *nidcpower.Session.self_cal()* method are written to hardware and used in the existing session and in all subsequent sessions until another call to the *nidcpower.Session.self_cal()* method is made. about supported devices. Default Value: *KEEP_IN_MEMORY*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific instruments within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container instruments to specify a subset.

Example: `my_session.instruments[ ... ].self_calibration_persistence`

To set/get on all instruments, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.self_calibration_persistence`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.SelfCalibrationPersistence
Permissions	read-write
Repeated Capabilities	instruments

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Advanced:Self-Calibration Persistence**
- C Attribute: **NIDCPOWER_ATTR_SELF_CALIBRATION_PERSISTENCE**

sense

nidcpower.Session.**sense**

Selects either local or remote sensing of the output voltage for the specified channel(s). Refer to the Local and Remote Sense topic in the NI DC Power Supplies and SMUs Help for more information about sensing voltage on supported channels and about devices that support local and/or remote sensing. Default Value: The default value is *LOCAL* if the device supports local sense. Otherwise, the default and only supported value is *REMOTE*.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sense`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sense`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Sense
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Measurement:Sense**
- C Attribute: **NIDCPOWER_ATTR_SENSE**

sequence_advance_trigger_type

nidcpower.Session.**sequence_advance_trigger_type**

Specifies the behavior of the Sequence Advance trigger. Default Value: *NONE*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_advance_trigger_type`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_advance_trigger_type`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TriggerType
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Sequence Advance Trigger:Trigger Type**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ADVANCE_TRIGGER_TYPE**

sequence_engine_done_event_output_behavior

`nidcpower.Session.`**`sequence_engine_done_event_output_behavior`**

Determines the event type's behavior when a corresponding trigger is received. If you set the Sequence Engine Done event output behavior to *PULSE*, a single pulse is transmitted. If you set the Sequence Engine Done event output behavior to *TOGGLE*, the output level toggles between low and high. The default value is *PULSE*.

Note: This property is not supported by all output terminals. This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_engine_done_event_output_behavior`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_engine_done_event_output_behavior`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventOutputBehavior
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Engine Done Event:Output Behavior**

- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ENGINE_DONE_EVENT_OUTPUT_BEHAVIOR**

sequence_engine_done_event_output_terminal

nidcpower.Session.**sequence_engine_done_event_output_terminal**

Specifies the output terminal for exporting the Sequence Engine Done Complete event. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_engine_done_event_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_engine_done_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Engine Done Event:Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ENGINE_DONE_EVENT_OUTPUT_TERMINAL**

sequence_engine_done_event_pulse_polarity

nidcpower.Session.**sequence_engine_done_event_pulse_polarity**

Specifies the behavior of the Sequence Engine Done event. Default Value: *HIGH*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_engine_done_event_pulse_polarity`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_engine_done_event_pulse_polarity`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Polarity
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Engine Done Event:Pulse:Polarity**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ENGINE_DONE_EVENT_PULSE_POLARITY**

sequence_engine_done_event_pulse_width

`nidcpower.Session.`**`sequence_engine_done_event_pulse_width`**

Specifies the width of the Sequence Engine Done event, in seconds. The minimum event pulse width value for PXI devices is 150 ns, and the minimum event pulse width value for PXI Express devices is 250 ns. The maximum event pulse width value for all devices is 1.6 microseconds. Valid Values: 1.5e-7 to 1.6e-6 seconds Default Value: The default value for PXI devices is 150 ns. The default value for PXI Express devices is 250 ns.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_engine_done_event_pulse_width`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_engine_done_event_pulse_width`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Engine Done Event:Pulse:Width**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ENGINE_DONE_EVENT_PULSE_WIDTH**

sequence_engine_done_event_toggle_initial_state

nidcpower.Session.**sequence_engine_done_event_toggle_initial_state**

Specifies the initial state of the Sequence Engine Done event when you set the *nidcpower.Session.sequence_engine_done_event_output_behavior* property to *TOGGLE*. For a Single Point mode acquisition, if you set the initial state to NIDCPOWER_VAL_LOW_STATE, the output is set to low at session commit. The output switches to high when the event occurs during the acquisition. If you set the initial state to NIDCPOWER_VAL_HIGH_STATE, the output is set to a high state at session commit. The output switches to low when the event occurs during the acquisition. For a Sequence mode operation, if you set the initial state to NIDCPOWER_VAL_LOW_STATE, the output is set to low at session commit. The output switches to high the first time an event occurs during the acquisition. The second time an event occurs, the output switches to low. This pattern repeats for any subsequent event occurrences. If you set the initial state to NIDCPOWER_VAL_HIGH_STATE, the output is set to high at session commit. The output switches to low on the first time the event occurs during the acquisition. The second time the event occurs, the output switches to high. This pattern repeats for any subsequent event occurrences. The default value is NIDCPOWER_VAL_LOW_STATE.

Note: This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com

Note: One or more of the referenced values are not in the Python API for this driver. Enums that only define values, or represent True/False, have been removed.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_engine_done_event_toggle_initial_state`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_engine_done_event_toggle_initial_state`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventToggleInitialState
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Engine Done Event:Toggle:Initial State**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ENGINE_DONE_EVENT_TOGGLE_INITIAL_STATE**

sequence_iteration_complete_event_output_behavior

nidcpower.Session.**sequence_iteration_complete_event_output_behavior**

Determines the event type's behavior when a corresponding trigger is received. If you set the Sequence Iteration Complete event output behavior to *PULSE*, a single pulse is transmitted. If you set the Sequence Iteration Complete event output behavior to *TOGGLE*, the output level toggles between low and high. The default value is *PULSE*.

Note: This property is not supported by all output terminals. This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_iteration_complete_event_output_behavior`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_iteration_complete_event_output_behavior`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventOutputBehavior
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Iteration Complete Event:Output Behavior**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ITERATION_COMPLETE_EVENT_OUTPUT_BEHAVIOR**

sequence_iteration_complete_event_output_terminal

nidcpower.Session.**sequence_iteration_complete_event_output_terminal**

Specifies the output terminal for exporting the Sequence Iteration Complete event. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_iteration_complete_event_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_iteration_complete_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Iteration Complete Event:Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ITERATION_COMPLETE_EVENT_OUTPUT_TERMINAL**

sequence_iteration_complete_event_pulse_polarity

`nidcpower.Session.`**`sequence_iteration_complete_event_pulse_polarity`**

Specifies the behavior of the Sequence Iteration Complete event. Default Value: *HIGH*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_iteration_complete_event_pulse_polarity`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_iteration_complete_event_pulse_polarity`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Polarity
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Iteration Complete Event:Pulse:Polarity**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ITERATION_COMPLETE_EVENT_PULSE_POLARITY**

sequence_iteration_complete_event_pulse_width

nidcpower.Session.**sequence_iteration_complete_event_pulse_width**

Specifies the width of the Sequence Iteration Complete event, in seconds. The minimum event pulse width value for PXI devices is 150 ns, and the minimum event pulse width value for PXI Express devices is 250 ns. The maximum event pulse width value for all devices is 1.6 microseconds. the NI DC Power Supplies and SMUs Help for information about supported devices. Valid Values: 1.5e-7 to 1.6e-6 seconds Default Value: The default value for PXI devices is 150 ns. The default value for PXI Express devices is 250 ns.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_iteration_complete_event_pulse_width`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_iteration_complete_event_pulse_width`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Iteration Complete Event:Pulse:Width**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ITERATION_COMPLETE_EVENT_PULSE_WIDTH**

---

sequence_iteration_complete_event_toggle_initial_state

nidcpower.Session.**sequence_iteration_complete_event_toggle_initial_state**

Specifies the initial state of the Sequence Iteration Complete event when you set the *nidcpower.Session.sequence_iteration_complete_event_output_behavior* property to *TOGGLE*. For a Single Point mode acquisition, if you set the initial state to `NIDCPOWER_VAL_LOW_STATE`, the output is set to low at session commit. The output switches to high when the event occurs during the acquisition. If you set the initial state to `NIDCPOWER_VAL_HIGH_STATE`, the output is set to a high state at session commit. The output switches to low when the event occurs during the acquisition. For a Sequence mode operation, if you set the initial state to `NIDCPOWER_VAL_LOW_STATE`, the output is set to low at session commit. The output switches to high the first time an event occurs during the acquisition. The second time an event occurs, the output switches to low. This pattern repeats for any subsequent event occurrences. If you set the initial state to `NIDCPOWER_VAL_HIGH_STATE`, the output is set to high at session commit. The output switches to low on the first time the event occurs

during the acquisition. The second time the event occurs, the output switches to high. This pattern repeats for any subsequent event occurrences. The default value is NIDCPOWER_VAL_LOW_STATE.

Note: This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com

Note: One or more of the referenced values are not in the Python API for this driver. Enums that only define values, or represent True/False, have been removed.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_iteration_complete_event_toggle_initial_state`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_iteration_complete_event_toggle_initial_state`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventToggleInitialState
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Sequence Iteration Complete Event:Toggle:Initial State**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_ITERATION_COMPLETE_EVENT_TOGGLE_INITIAL_ST**

sequence_loop_count

nidcpower.Session.**sequence_loop_count**

Specifies the number of times a sequence is run after initiation. Refer to the Sequence Source Mode topic in the NI DC Power Supplies and SMUs Help for more information about the sequence loop count. When the *nidcpower.Session.sequence_loop_count_is_finite* property is set to False, the *nidcpower.Session.sequence_loop_count* property is ignored. Valid Range: 1 to 2147483647 Default Value: 1

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_loop_count`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_loop_count`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Sequence Loop Count**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_LOOP_COUNT**

sequence_loop_count_is_finite

`nidcpower.Session.`**`sequence_loop_count_is_finite`**

Specifies whether a sequence should repeat indefinitely. Refer to the Sequence Source Mode topic in the NI DC Power Supplies and SMUs Help for more information about infinite sequencing. When the *nidcpower.Session.sequence_loop_count_is_finite* property is set to False, the *nidcpower.Session.sequence_loop_count* property is ignored. Default Value: True

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_loop_count_is_finite`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_loop_count_is_finite`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Sequence Loop Count Is Finite**
- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_LOOP_COUNT_IS_FINITE**

sequence_step_delta_time

nidcpower.Session.**sequence_step_delta_time**

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_step_delta_time`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_step_delta_time`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_STEP_DELTA_TIME**

sequence_step_delta_time_enabled

nidcpower.Session.**sequence_step_delta_time_enabled**

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].sequence_step_delta_time_enabled`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.sequence_step_delta_time_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- C Attribute: **NIDCPOWER_ATTR_SEQUENCE_STEP_DELTA_TIME_ENABLED**

serial_number

nidcpower.Session.**serial_number**

Contains the serial number for the device you are currently using.

Tip: This property can be set/get on specific instruments within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container instruments to specify a subset.

Example: `my_session.instruments[ ... ].serial_number`

To set/get on all instruments, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.serial_number`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	instruments

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Serial Number**
- C Attribute: **NIDCPOWER_ATTR_SERIAL_NUMBER**

shutdown_trigger_type

nidcpower.Session.**shutdown_trigger_type**

Specifies the behavior of the Shutdown trigger. Default Value: *NONE*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].shutdown_trigger_type`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.shutdown_trigger_type`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TriggerType
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Shutdown Trigger:Trigger Type**
- C Attribute: **NIDCPOWER_ATTR_SHUTDOWN_TRIGGER_TYPE**

simulate

`nidcpower.Session.`**`simulate`**

Specifies whether to simulate NI-DCPower I/O operations. True specifies that operation is simulated. Default Value: False

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:User Options:Simulate**
- C Attribute: **NIDCPOWER_ATTR_SIMULATE**

source_complete_event_output_behavior

`nidcpower.Session.`**`source_complete_event_output_behavior`**

Determines the event type's behavior when a corresponding trigger is received. If you set the Source Complete event output behavior to *PULSE*, a single pulse is transmitted. If you set the Source Complete event output behavior to *TOGGLE*, the output level toggles between low and high. The default value is *PULSE*.

Note: This property is not supported by all output terminals. This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_complete_event_output_behavior`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_complete_event_output_behavior`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventOutputBehavior
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Source Complete Event:Output Behavior**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_COMPLETE_EVENT_OUTPUT_BEHAVIOR**

source_complete_event_output_terminal

nidcpower.Session.**source_complete_event_output_terminal**

Specifies the output terminal for exporting the Source Complete event. Output terminals can be specified in one of two ways. If the device is named Dev1 and your terminal is PXI_Trig0, you can specify the terminal with the fully qualified terminal name, /Dev1/PXI_Trig0, or with the shortened terminal name, PXI_Trig0.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_complete_event_output_terminal`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_complete_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Source Complete Event:Output Terminal**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_COMPLETE_EVENT_OUTPUT_TERMINAL**

source_complete_event_pulse_polarity

nidcpower.Session.**source_complete_event_pulse_polarity**

Specifies the behavior of the Source Complete event. Default Value: *HIGH*

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_complete_event_pulse_polarity`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_complete_event_pulse_polarity`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Polarity
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Source Complete Event:Pulse:Polarity**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_COMPLETE_EVENT_PULSE_POLARITY**

source_complete_event_pulse_width

nidcpower.Session.**source_complete_event_pulse_width**

Specifies the width of the Source Complete event, in seconds. The minimum event pulse width value for PXI devices is 150 ns, and the minimum event pulse width value for PXI Express devices is 250 ns. The maximum event pulse width value for all devices is 1.6 microseconds Valid Values: 1.5e-7 to 1.6e-6 seconds Default Value: The default value for PXI devices is 150 ns. The default value for PXI Express devices is 250 ns.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_complete_event_pulse_width`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_complete_event_pulse_width`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Source Complete Event:Pulse:Width**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_COMPLETE_EVENT_PULSE_WIDTH**

---

source_complete_event_toggle_initial_state

nidcpower.Session.**source_complete_event_toggle_initial_state**

Specifies the initial state of the Source Complete event when you set the *nidcpower.Session.source_complete_event_output_behavior* property to *TOGGLE*. For a Single Point mode acquisition, if you set the initial state to `NIDCPOWER_VAL_LOW_STATE`, the output is set to low at session commit. The output switches to high when the event occurs during the acquisition. If you set the initial state to `NIDCPOWER_VAL_HIGH_STATE`, the output is set to a high state at session commit. The output switches to low when the event occurs during the acquisition. For a Sequence mode operation, if you set the initial state to `NIDCPOWER_VAL_LOW_STATE`, the output is set to low at session commit. The output switches to high the first time an event occurs during the acquisition. The second time an event occurs, the output switches to low. This pattern repeats for any subsequent event occurrences.

If you set the initial state to NIDCPOWER_VAL_HIGH_STATE, the output is set to high at session commit. The output switches to low on the first time the event occurs during the acquisition. The second time the event occurs, the output switches to high. This pattern repeats for any subsequent event occurrences. The default value is NIDCPOWER_VAL_LOW_STATE.

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices and terminals, search Supported Properties by Device on ni.com

---

Note: One or more of the referenced values are not in the Python API for this driver. Enums that only define values, or represent True/False, have been removed.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_complete_event_toggle_initial_state`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_complete_event_toggle_initial_state`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.EventToggleInitialState
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Events:Source Complete Event:Toggle:Initial State**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_COMPLETE_EVENT_TOGGLE_INITIAL_STATE**

---

source_delay

`nidcpower.Session.`**`source_delay`**

Determines when, in seconds, the device generates the Source Complete event, potentially starting a measurement if the *nidcpower.Session.measure_when* property is set to *AUTOMATICALLY_AFTER_SOURCE_COMPLETE*. Refer to the Single Point Source Mode and Sequence Source Mode topics for more information. Valid Values: The PXIe-4051 supports values from 0 to 39 seconds. The PXIe-4147 supports values from 0 to 26.5 seconds. The PXIe-4151 supports values from 0 to 42 seconds. The PXIe-4162/4163 and PXIe-4190 support values from 0 to 23 seconds. All other supported instruments support values from 0 to 167 seconds. Default Value: 0.01667 seconds

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_delay`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_delay`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Advanced:Source Delay**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_DELAY**

source_mode

`nidcpower.Session.`**`source_mode`**

Specifies whether to run a single output point or a sequence. Refer to the Single Point Source Mode and Sequence Source Mode topics in the NI DC Power Supplies and SMUs Help for more information about source modes. Default value: *SINGLE_POINT*

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_mode`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_mode`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.SourceMode
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Source Mode**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_MODE**

source_trigger_type

nidcpower.Session.**source_trigger_type**

Specifies the behavior of the Source trigger. Default Value: *NONE*

Note: NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].source_trigger_type`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.source_trigger_type`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TriggerType
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Source Trigger:Trigger Type**
- C Attribute: **NIDCPOWER_ATTR_SOURCE_TRIGGER_TYPE**

specific_driver_description

nidcpower.Session.**specific_driver_description**

Contains a brief description of the specific driver.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Driver Identification:Description**
- C Attribute: **NIDCPOWER_ATTR_SPECIFIC_DRIVER_DESCRIPTION**

specific_driver_prefix

nidcpower.Session.**specific_driver_prefix**

Contains the prefix for NI-DCPower. The name of each user-callable method in NI-DCPower begins with this prefix.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Driver Identification:Driver Prefix**
- C Attribute: **NIDCPOWER_ATTR_SPECIFIC_DRIVER_PREFIX**

specific_driver_revision

nidcpower.Session.**specific_driver_revision**

Contains additional version information about NI-DCPower.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Driver Identification:Revision**
- C Attribute: **NIDCPOWER_ATTR_SPECIFIC_DRIVER_REVISION**

specific_driver_vendor

nidcpower.Session.**specific_driver_vendor**

Contains the name of the vendor that supplies NI-DCPower.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Driver Identification:Driver Vendor**
- C Attribute: **NIDCPOWER_ATTR_SPECIFIC_DRIVER_VENDOR**

start_trigger_type

nidcpower.Session.**start_trigger_type**

Specifies the behavior of the Start trigger. Default Value: *NONE*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].start_trigger_type`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.start_trigger_type`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TriggerType
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Triggers:Start Trigger:Trigger Type**
- C Attribute: **NIDCPOWER_ATTR_START_TRIGGER_TYPE**

supported_instrument_models

nidcpower.Session.**supported_instrument_models**

Contains a comma-separated (,) list of supported NI-DCPower device models.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Inherent IVI Attributes:Driver Capabilities:Supported Instrument Models**
- C Attribute: **NIDCPOWER_ATTR_SUPPORTED_INSTRUMENT_MODELS**

transient_response

nidcpower.Session.**transient_response**

Specifies the transient response. Refer to the Transient Response topic in the NI DC Power Supplies and SMUs Help for more information about transient response. Default Value: *NORMAL*

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].transient_response`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.transient_response`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TransientResponse
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Transient Response**
- C Attribute: **NIDCPOWER_ATTR_TRANSIENT_RESPONSE**

voltage_compensation_frequency

nidcpower.Session.**voltage_compensation_frequency**

The frequency at which a pole-zero pair is added to the system when the channel is in Constant Voltage mode. Default value: Determined by the value of the *NORMAL* setting of the *nidcpower.Session.transient_response* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_compensation_frequency`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_compensation_frequency`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Custom Transient Response:Voltage:Compensation Frequency**

- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_COMPENSATION_FREQUENCY**

voltage_gain_bandwidth

nidcpower.Session.**voltage_gain_bandwidth**

The frequency at which the unloaded loop gain extrapolates to 0 dB in the absence of additional poles and zeroes. This property takes effect when the channel is in Constant Voltage mode. Default Value: Determined by the value of the *NORMAL* setting of the *nidcpower.Session.transient_response* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_gain_bandwidth`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_gain_bandwidth`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Custom Transient Response:Voltage:Gain Bandwidth**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_GAIN_BANDWIDTH**

voltage_level

nidcpower.Session.**voltage_level**

Specifies the voltage level, in volts, that the device attempts to generate on the specified channel(s). This property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE*.

Valid Values: The valid values for this property are defined by the values you specify for the *nidcpower.Session.voltage_level_range* property.

Note: The channel must be enabled for the specified voltage level to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_level`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_level`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Voltage Level**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LEVEL**

voltage_level_autorange

`nidcpower.Session.`**`voltage_level_autorange`**

Specifies whether NI-DCPower automatically selects the voltage level range based on the desired voltage level for the specified channel(s). If you set this property to *ON*, NI-DCPower ignores any changes you make to the *nidcpower.Session.voltage_level_range* property. If you change the *nidcpower.Session.voltage_level_autorange* property from *ON* to *OFF*, NI-DCPower retains the last value the *nidcpower.Session.voltage_level_range* property was set to (or the default value if the property was never set) and uses that value as the voltage level range. Query the *nidcpower.Session.voltage_level_range* property by using the `nidcpower.Session._get_attribute_vi_int32()` method for information about which range NI-DCPower automatically selects. The *nidcpower.Session.voltage_level_autorange* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE*. Default Value: *OFF*

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_level_autorange`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_level_autorange`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Voltage Level Autorange**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LEVEL_AUTORANGE**

voltage_level_range

nidcpower.Session.**voltage_level_range**

Specifies the voltage level range, in volts, for the specified channel(s). The range defines the valid values to which the voltage level can be set. Use the *nidcpower.Session.voltage_level_autorange* property to enable automatic selection of the voltage level range. The *nidcpower.Session.voltage_level_range* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_VOLTAGE*.

For valid ranges, refer to the specifications for your instrument.

Note: The channel must be enabled for the specified voltage level range to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_level_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_level_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Voltage:Voltage Level Range**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE**

voltage_limit

nidcpower.Session.**voltage_limit**

Specifies the voltage limit, in volts, that the output cannot exceed when generating the desired current level on the specified channels. This property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_CURRENT* and the *nidcpower.Session.compliance_limit_symmetry* property is set to *SYMMETRIC*.

Valid Values: The valid values for this property are defined by the values to which the *nidcpower.Session.voltage_limit_range* property is set.

Note: The channel must be enabled for the specified current level to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

NI-DCPower uses the terms "source" and "output". However, while sinking with electronic loads and SMUs these correspond to "sinking" and "input", respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_limit`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_limit`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Voltage Limit**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LIMIT**

---

voltage_limit_autorange

nidcpower.Session.**voltage_limit_autorange**

Specifies whether NI-DCPower automatically selects the voltage limit range based on the desired voltage limit for the specified channel(s). If this property is set to *ON*, NI-DCPower ignores any changes you make to the *nidcpower.Session.voltage_limit_range* property. If you change the *nidcpower.Session.voltage_limit_autorange* property from *ON* to *OFF*, NI-DCPower retains the last value the *nidcpower.Session.voltage_limit_range* property was set to (or the default value if the property was never set) and uses that value as the voltage limit range. Query

the *nidcpower.Session.voltage_limit_range* property by using the nidcpower.Session._get_attribute_vi_int32() method to find out which range NI-DCPower automatically selects. The *nidcpower.Session.voltage_limit_autorange* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_CURRENT*. Default Value: *OFF*

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: my_session.channels[...].voltage_limit_autorange

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: my_session.voltage_limit_autorange

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Voltage Limit Autorange**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LIMIT_AUTORANGE**

---

voltage_limit_high

nidcpower.Session.**voltage_limit_high**

Specifies the maximum voltage, in volts, that the output can produce when generating the desired current on the specified channel(s). This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *DC_CURRENT*. You must also specify a *nidcpower.Session.voltage_limit_low* to complete the asymmetric range. **Valid Values:** [1% of *nidcpower.Session.voltage_limit_range*, *nidcpower.Session.voltage_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

---

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True.

NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: my_session.channels[...].voltage_limit_high

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_limit_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Voltage Limit High**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LIMIT_HIGH**

---

voltage_limit_low

`nidcpower.Session.`**`voltage_limit_low`**

Specifies the minimum voltage, in volts, that the output can produce when generating the desired current on the specified channel(s). This property is applicable only if the *nidcpower.Session.compliance_limit_symmetry* property is set to *ASYMMETRIC* and the *nidcpower.Session.output_function* property is set to *DC_CURRENT*. You must also specify a *nidcpower.Session.voltage_limit_high* to complete the asymmetric range. **Valid Values:** [-*nidcpower.Session.voltage_limit_range*, -1% of *nidcpower.Session.voltage_limit_range*] The range bounded by the limit high and limit low must include zero. **Default Value:** Search ni.com for Supported Properties by Device for the default value by device. **Related Topics:** Ranges; Changing Ranges; Overranging

---

Note: The limit may be extended beyond the selected limit range if the *nidcpower.Session.overranging_enabled* property is set to True.

NI-DCPower uses the terms “source” and “output”. However, while sinking with electronic loads and SMUs these correspond to “sinking” and “input”, respectively.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_limit_low`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_limit_low`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Voltage Limit Low**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LIMIT_LOW**

voltage_limit_range

nidcpower.Session.**voltage_limit_range**

Specifies the voltage limit range, in volts, for the specified channel(s). The range defines the valid values to which the voltage limit can be set. Use the *nidcpower.Session.voltage_limit_autorange* property to enable automatic selection of the voltage limit range. The *nidcpower.Session.voltage_limit_range* property is applicable only if the *nidcpower.Session.output_function* property is set to *DC_CURRENT*.

For valid ranges, refer to the specifications for your instrument.

Note: The channel must be enabled for the specified voltage limit range to take effect. Refer to the *nidcpower.Session.output_enabled* property for more information about enabling the channel.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_limit_range`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_limit_range`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:DC Current:Voltage Limit Range**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE**

voltage_pole_zero_ratio

nidcpower.Session.**voltage_pole_zero_ratio**

The ratio of the pole frequency to the zero frequency when the channel is in Constant Voltage mode. Default value: Determined by the value of the *NORMAL* setting of the *nidcpower.Session.transient_response* property.

Note: This property is not supported on all devices. For more information about supported devices, search ni.com for Supported Properties by Device.

Tip: This property can be set/get on specific channels within your *nidcpower.Session* instance. Use Python index notation on the repeated capabilities container channels to specify a subset.

Example: `my_session.channels[ ... ].voltage_pole_zero_ratio`

To set/get on all channels, you can call the property directly on the *nidcpower.Session*.

Example: `my_session.voltage_pole_zero_ratio`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Source:Custom Transient Response:Voltage:Pole-Zero Ratio**
- C Attribute: **NIDCPOWER_ATTR_VOLTAGE_POLE_ZERO_RATIO**

Session

- *create_advanced_sequence_commit_step*
- *create_advanced_sequence_step*
- *delete_advanced_sequence*
- *disable*
- *export_attribute_configuration_buffer*
- *export_attribute_configuration_file*
- *fetch_multiple*
- *fetch_multiple_lcr*
- *get_channel_name*
- *get_channel_names*
- *get_ext_cal_last_date_and_time*
- *get_ext_cal_last_temp*
- *get_ext_cal_recommended_interval*
- *get_lcr_compensation_data*
- *get_lcr_compensation_last_date_and_time*
- *get_lcr_custom_cable_compensation_data*
- *get_self_cal_last_date_and_time*
- *get_self_cal_last_temp*
- *import_attribute_configuration_buffer*
- *import_attribute_configuration_file*
- *initiate*
- *lock*
- *measure*
- *measure_multiple*
- *measure_multiple_lcr*
- *perform_lcr_load_compensation*
- *perform_lcr_open_compensation*
- *perform_lcr_open_custom_cable_compensation*
- *perform_lcr_short_compensation*
- *perform_lcr_short_custom_cable_compensation*
- *query_in_compliance*
- *query_latched_output_cutoff_state*
- *query_max_current_limit*
- *query_max_voltage_level*
- *query_min_current_limit*

- *current_compensation_frequency*
- *current_gain_bandwidth*
- *current_level*
- *current_level_autorange*
- *current_level_falling_slew_rate*
- *current_level_range*
- *current_level_rising_slew_rate*
- *current_limit*
- *current_limit_autorange*
- *current_limit_behavior*
- *current_limit_high*
- *current_limit_low*
- *current_limit_range*
- *current_pole_zero_ratio*
- *dc_noise_rejection*
- *digital_edge_measure_trigger_input_terminal*
- *digital_edge_pulse_trigger_input_terminal*
- *digital_edge_sequence_advance_trigger_input_terminal*
- *digital_edge_shutdown_trigger_input_terminal*
- *digital_edge_source_trigger_input_terminal*
- *digital_edge_start_trigger_input_terminal*
- *driver_setup*
- *exported_measure_trigger_output_terminal*
- *exported_pulse_trigger_output_terminal*
- *exported_sequence_advance_trigger_output_terminal*
- *exported_source_trigger_output_terminal*
- *exported_start_trigger_output_terminal*
- *fetch_backlog*
- *instrument_firmware_revision*
- *instrument_manufacturer*
- *instrument_mode*
- *instrument_model*
- *interlock_input_open*
- *io_resource_descriptor*
- *isolation_state*

- *lcr_actual_load_reactance*
- *lcr_actual_load_resistance*
- *lcr_ac_dither_enabled*
- *lcr_ac_electrical_cable_length_delay*
- *lcr_automatic_level_control*
- *lcr_current_amplitude*
- *lcr_current_range*
- *lcr_custom_measurement_time*
- *lcr_dc_bias_automatic_level_control*
- *lcr_dc_bias_current_level*
- *lcr_dc_bias_current_range*
- *lcr_dc_bias_source*
- *lcr_dc_bias_transient_response*
- *lcr_dc_bias_voltage_level*
- *lcr_dc_bias_voltage_range*
- *lcr_frequency*
- *lcr_impedance_auto_range*
- *lcr_impedance_range*
- *lcr_impedance_range_source*
- *lcr_load_capacitance*
- *lcr_load_compensation_enabled*
- *lcr_load_inductance*
- *lcr_load_resistance*
- *lcr_measured_load_reactance*
- *lcr_measured_load_resistance*
- *lcr_measurement_time*
- *lcr_open_compensation_enabled*
- *lcr_open_conductance*
- *lcr_open_short_load_compensation_data_source*
- *lcr_open_susceptance*
- *lcr_short_compensation_enabled*
- *lcr_short_custom_cable_compensation_enabled*
- *lcr_short_reactance*
- *lcr_short_resistance*
- *lcr_source_aperture_time*

- *lcr_source_delay_mode*
- *lcr_stimulus_function*
- *lcr_voltage_amplitude*
- *lcr_voltage_range*
- *logical_name*
- *measure_buffer_size*
- *measure_complete_event_delay*
- *measure_complete_event_output_behavior*
- *measure_complete_event_output_terminal*
- *measure_complete_event_pulse_polarity*
- *measure_complete_event_pulse_width*
- *measure_complete_event_toggle_initial_state*
- *measure_record_delta_time*
- *measure_record_length*
- *measure_record_length_is_finite*
- *measure_trigger_type*
- *measure_when*
- *merged_channels*
- *output_capacitance*
- *output_connected*
- *output_cutoff_current_change_limit_high*
- *output_cutoff_current_change_limit_low*
- *output_cutoff_current_measure_limit_high*
- *output_cutoff_current_measure_limit_low*
- *output_cutoff_current_overrange_enabled*
- *output_cutoff_delay*
- *output_cutoff_enabled*
- *output_cutoff_voltage_change_limit_high*
- *output_cutoff_voltage_change_limit_low*
- *output_cutoff_voltage_measure_limit_high*
- *output_cutoff_voltage_measure_limit_low*
- *output_cutoff_voltage_output_limit_high*
- *output_cutoff_voltage_output_limit_low*
- *output_enabled*
- *output_function*

- *output_resistance*
- *overranging_enabled*
- *ovp_enabled*
- *ovp_limit*
- *power_allocation_mode*
- *power_line_frequency*
- *power_source*
- *power_source_in_use*
- *pulse_bias_current_level*
- *pulse_bias_current_limit*
- *pulse_bias_current_limit_high*
- *pulse_bias_current_limit_low*
- *pulse_bias_delay*
- *pulse_bias_voltage_level*
- *pulse_bias_voltage_limit*
- *pulse_bias_voltage_limit_high*
- *pulse_bias_voltage_limit_low*
- *pulse_complete_event_output_terminal*
- *pulse_complete_event_pulse_polarity*
- *pulse_complete_event_pulse_width*
- *pulse_current_level*
- *pulse_current_level_range*
- *pulse_current_limit*
- *pulse_current_limit_high*
- *pulse_current_limit_low*
- *pulse_current_limit_range*
- *pulse_off_time*
- *pulse_on_time*
- *pulse_trigger_type*
- *pulse_voltage_level*
- *pulse_voltage_level_range*
- *pulse_voltage_limit*
- *pulse_voltage_limit_high*
- *pulse_voltage_limit_low*
- *pulse_voltage_limit_range*

- *query_instrument_status*
- *ready_for_pulse_trigger_event_output_terminal*
- *ready_for_pulse_trigger_event_pulse_polarity*
- *ready_for_pulse_trigger_event_pulse_width*
- *requested_power_allocation*
- *reset_average_before_measurement*
- *samples_to_average*
- *self_calibration_persistence*
- *sense*
- *sequence_advance_trigger_type*
- *sequence_engine_done_event_output_behavior*
- *sequence_engine_done_event_output_terminal*
- *sequence_engine_done_event_pulse_polarity*
- *sequence_engine_done_event_pulse_width*
- *sequence_engine_done_event_toggle_initial_state*
- *sequence_iteration_complete_event_output_behavior*
- *sequence_iteration_complete_event_output_terminal*
- *sequence_iteration_complete_event_pulse_polarity*
- *sequence_iteration_complete_event_pulse_width*
- *sequence_iteration_complete_event_toggle_initial_state*
- *sequence_loop_count*
- *sequence_loop_count_is_finite*
- *sequence_step_delta_time*
- *sequence_step_delta_time_enabled*
- *serial_number*
- *shutdown_trigger_type*
- *simulate*
- *source_complete_event_output_behavior*
- *source_complete_event_output_terminal*
- *source_complete_event_pulse_polarity*
- *source_complete_event_pulse_width*
- *source_complete_event_toggle_initial_state*
- *source_delay*
- *source_mode*
- *source_trigger_type*

- *specific_driver_description*
- *specific_driver_prefix*
- *specific_driver_revision*
- *specific_driver_vendor*
- *start_trigger_type*
- *supported_instrument_models*
- *transient_response*
- *voltage_compensation_frequency*
- *voltage_gain_bandwidth*
- *voltage_level*
- *voltage_level_autorange*
- *voltage_level_range*
- *voltage_limit*
- *voltage_limit_autorange*
- *voltage_limit_high*
- *voltage_limit_low*
- *voltage_limit_range*
- *voltage_pole_zero_ratio*

Repeated Capabilities

Repeated capabilities attributes are used to set the *channel_string* parameter to the underlying driver function call. This can be the actual function based on the `Session` method being called, or it can be the appropriate Get/Set Attribute function, such as `niDCPower_SetAttributeViInt32()`.

Repeated capabilities attributes use the indexing operator `[]` to indicate the repeated capabilities. The parameter can be a string, list, tuple, or slice (range). Each element of those can be a string or an integer. If it is a string, you can indicate a range using the same format as the driver: `'0-2'` or `'0:2'`

Some repeated capabilities use a prefix before the number and this is optional

channels

nidcpower.Session.**channels**

```
session.channels['0-2'].channel_enabled = True
```

passes a string of `'0, 1, 2'` to the set attribute function.

instruments

nidcpower.Session.**instruments**

```
session.instruments['0-2'].channel_enabled = True
```

passes a string of `'0, 1, 2'` to the set attribute function.

Enums

Enums used in NI-DCPower

ApertureTimeAutoMode

class nidcpower.**ApertureTimeAutoMode**

OFF

Disables automatic aperture time scaling. The *nidcpower.Session.aperture_time* property specifies the aperture time for all ranges.

SHORT

Prioritizes measurement speed over measurement accuracy by quickly scaling down aperture time in larger current ranges. The *nidcpower.Session.aperture_time* property specifies the aperture time for the minimum range.

NORMAL

Balances measurement accuracy and speed by scaling down aperture time in larger current ranges. The *nidcpower.Session.aperture_time* property specifies the aperture time for the minimum range.

LONG

Prioritizes accuracy while still decreasing measurement time by slowly scaling down aperture time in larger current ranges. The *nidcpower.Session.aperture_time* property specifies the aperture time for the minimum range.

ApertureTimeUnits

class nidcpower.**ApertureTimeUnits**

SECONDS

Specifies aperture time in seconds.

POWER_LINE_CYCLES

Specifies aperture time in power line cycles (PLCs).

AutoZero

class nidcpower.**AutoZero**

OFF

Disables auto zero.

ONCE

Makes zero conversions following the first measurement after initiating the device. The device uses these zero conversions for the preceding measurement and future measurements until the device is reinitiated.

ON

Makes zero conversions for every measurement.

AutorangeApertureTimeMode

class nidcpower.**AutorangeApertureTimeMode**

AUTO

NI-DCPower optimizes the aperture time for the autorange algorithm based on the module range.

CUSTOM

The user specifies a minimum aperture time for the algorithm using the *nidcpower.Session.autorange_minimum_aperture_time* property and the corresponding *nidcpower.Session.autorange_minimum_aperture_time_units* property.

AutorangeBehavior

class nidcpower.**AutorangeBehavior**

UP_TO_LIMIT_THEN_DOWN

Go to limit range then range down as needed until measured value is within thresholds.

UP

go up one range when the upper threshold is reached.

UP_AND_DOWN

go up or down one range when the upper/lower threshold is reached.

AutorangeThresholdMode

class nidcpower.**AutorangeThresholdMode**

NORMAL

Thresholds are selected based on a balance between accuracy and hysteresis.

FAST_STEP

Optimized for faster changes in the measured signal. Thresholds are configured to be a smaller percentage of the range.

HIGH_HYSTERESIS

Optimized for noisy signals to minimize frequent and unpredictable range changes. Thresholds are configured to be a larger percentage of the range.

MEDIUM_HYSTERESIS

Optimized for noisy signals to minimize frequent and unpredictable range changes. Thresholds are configured to be a medium percentage of the range.

HOLD

Attempt to maintain the active range. Thresholds will favor the active range.

CableLength

class nidcpower.**CableLength**

ZERO_M

Uses predefined cable compensation data for a 0m cable (direct connection).

NI_STANDARD_0_5M

Uses predefined cable compensation data for an NI standard 0.5m coaxial cable.

NI_STANDARD_1M

Uses predefined cable compensation data for an NI standard 1m coaxial cable.

NI_STANDARD_2M

Uses predefined cable compensation data for an NI standard 2m coaxial cable.

NI_STANDARD_4M

Uses predefined cable compensation data for an NI standard 4m coaxial cable.

NI_STANDARD_TRIAXIAL_1M

Uses predefined cable compensation data for an NI standard 1m triaxial cable.

NI_STANDARD_TRIAXIAL_2M

Uses predefined cable compensation data for an NI standard 2m triaxial cable.

NI_STANDARD_TRIAXIAL_4M

Uses predefined cable compensation data for an NI standard 4m triaxial cable.

CUSTOM_ONBOARD_STORAGE

Uses previously generated custom cable compensation data from onboard storage. Only the most recently performed compensation data for each custom cable compensation type (open, short) is stored.

CUSTOM_AS_CONFIGURED

Uses the custom cable compensation data supplied to *nidcpower.Session.configure_lcr_custom_cable_compensation()*. Use this option to manage multiple sets of custom cable compensation data.

ComplianceLimitSymmetry

class nidcpower.**ComplianceLimitSymmetry**

SYMMETRIC

Compliance limits are specified symmetrically about 0.

ASYMMETRIC

Compliance limits can be specified asymmetrically with respect to 0.

ConductionVoltageMode

class nidcpower.**ConductionVoltageMode**

AUTOMATIC

The conduction voltage feature is only enabled when you set the *nidcpower.Session.output_function* property to *DC_CURRENT*.

ENABLED

The conduction voltage feature is enabled.

DISABLED

The conduction voltage feature is disabled.

CurrentLimitBehavior

class nidcpower.**CurrentLimitBehavior**

REGULATE

The channel acts to restrict the output current to the value of the Current Limit property when the actual output on the channel reaches or exceeds that value.

TRIP

The channel disables the output when the actual output current on the channel reaches or exceeds the value of the Current Limit property.

DCNoiseRejection

class nidcpower.**DCNoiseRejection**

SECOND_ORDER

Second-order rejection of DC noise.

NORMAL

Normal rejection of DC noise.

Event

class nidcpower.**Event**

SOURCE_COMPLETE
Specifies the Source Complete event.

MEASURE_COMPLETE
Specifies the Measure Complete event.

SEQUENCE_ITERATION_COMPLETE
Specifies the Sequence Iteration Complete event.

SEQUENCE_ENGINE_DONE
Specifies the Sequence Engine Done event.

PULSE_COMPLETE
Specifies the Pulse Complete event.

READY_FOR_PULSE_TRIGGER
Specifies the Ready for Pulse Trigger event.

EventOutputBehavior

class nidcpower.**EventOutputBehavior**

PULSE
Output generates a pulse when the event is triggered.

TOGGLE
Output toggles state when the event is triggered.

EventToggleInitialState

class nidcpower.**EventToggleInitialState**

LOW
The initial state is low.

HIGH
The initial state is high.

InstrumentMode

class nidcpower.**InstrumentMode**

SMU_PS
The channel operates as an SMU/power supply.

LCR
The channel operates as an LCR meter.

E_LOAD
The channel operates as an electronic load (E-Load).

LCRCompensationType

class nidcpower.**LCRCompensationType**

OPEN

Open LCR compensation.

SHORT

Short LCR compensation.

LOAD

Load LCR compensation.

OPEN_CUSTOM_CABLE

Open custom cable compensation.

SHORT_CUSTOM_CABLE

Short custom cable compensation.

LCRDCBiasSource

class nidcpower.**LCRDCBiasSource**

OFF

Disables DC bias in LCR mode.

VOLTAGE

Applies a constant voltage bias, as defined by the *nidcpower.Session.lcr_dc_bias_voltage_level* property.

CURRENT

Applies a constant current bias, as defined by the *nidcpower.Session.lcr_dc_bias_current_level* property.

LCRDCBiasTransientResponse

class nidcpower.**LCRDCBiasTransientResponse**

NORMAL

NI-DCPower automatically applies transient response values for DC bias.

CUSTOM

NI-DCPower applies the transient response that you set manually with *nidcpower.Session.transient_response* for DC bias. Search ni.com for information on configuring transient response.

LCRImpedanceRangeSource

class nidcpower.**LCRImpedanceRangeSource**

IMPEDANCE_RANGE

Uses the impedance range you specify with the *nidcpower.Session.lcr_impedance_range* property.

LOAD_CONFIGURATION

Computes the impedance range to select based on the values you supply to the *nidcpower.Session.lcr_load_resistance*, *nidcpower.Session.lcr_load_inductance*, and *nidcpower.Session.lcr_load_capacitance* properties. NI-DCPower uses a series model of load resistance, load inductance, and load capacitance to compute the impedance range.

LCRMeasurementTime

class nidcpower.**LCRMeasurementTime**

SHORT

Uses a short aperture time for LCR measurements.

MEDIUM

Uses a medium aperture time for LCR measurements.

LONG

Uses a long aperture time for LCR measurements.

CUSTOM

Uses a custom aperture time for LCR measurements as specified by the *nidcpower.Session.lcr_custom_measurement_time* property.

LCROpenShortLoadCompensationDataSource

class nidcpower.**LCROpenShortLoadCompensationDataSource**

ONBOARD_STORAGE

Uses previously generated LCR compensation data. Only the most recently performed compensation data for each LCR compensation type (open, short, and load) is stored.

AS_DEFINED

Uses the LCR compensation data represented by the relevant LCR compensation properties as generated by *nidcpower.Session.perform_lcr_open_compensation()*, *nidcpower.Session.perform_lcr_short_compensation()*, and *nidcpower.Session.perform_lcr_load_compensation()*. Use this option to manage multiple sets of LCR compensation data. This option applies compensation data from the following properties: *nidcpower.Session.lcr_open_conductance*, *nidcpower.Session.lcr_open_susceptance*, *nidcpower.Session.lcr_short_resistance*, *nidcpower.Session.lcr_short_reactance*, *nidcpower.Session.lcr_measured_load_resistance*, *nidcpower.Session.lcr_measured_load_reactance*, *nidcpower.Session.lcr_actual_load_resistance*, *nidcpower.Session.lcr_actual_load_reactance*.

AS_CONFIGURED

Uses the LCR compensation data supplied to *nidcpower.Session.configure_lcr_compensation()*. Use this option to manage multiple sets of LCR compensation data.

LCRReferenceValueType

class nidcpower.**LCRReferenceValueType**

IMPEDANCE

The actual impedance, comprising real resistance and imaginary reactance, of your DUT. Supply resistance, in ohms, to reference value A; supply reactance, in ohms, to reference value B.

IDEAL_CAPACITANCE

The ideal capacitance of your DUT. Supply capacitance, in farads, to reference value A.

IDEAL_INDUCTANCE

The ideal inductance of your DUT. Supply inductance, in henrys, to reference value A.

IDEAL_RESISTANCE

The ideal resistance of your DUT. Supply resistance, in ohms, to reference value A.

LCRSourceDelayMode

class nidcpower.**LCRSourceDelayMode**

AUTOMATIC

NI-DCPower automatically applies source delay of sufficient duration to account for settling time.

MANUAL

NI-DCPower applies the source delay that you set manually with *nidcpower.Session.source_delay*. You can use this option to set a shorter delay to reduce measurement time at the possible expense of measurement accuracy.

LCRStimulusFunction

class nidcpower.**LCRStimulusFunction**

VOLTAGE

Applies an AC voltage for LCR stimulus.

CURRENT

Applies an AC current for LCR stimulus.

MeasureWhen

class nidcpower.**MeasureWhen**

AUTOMATICALLY_AFTER_SOURCE_COMPLETE

Acquires a measurement after each Source Complete event completes.

ON_DEMAND

Acquires a measurement when the *nidcpower.Session.measure()* method or *nidcpower.Session.measure_multiple()* method is called.

ON_MEASURE_TRIGGER

Acquires a measurement when a Measure trigger is received.

MeasurementTypes

class nidcpower.**MeasurementTypes**

CURRENT

The device measures current.

VOLTAGE

The device measures voltage.

OutputCapacitance

class nidcpower.**OutputCapacitance**

LOW

Output Capacitance is low.

HIGH

Output Capacitance is high.

OutputCutoffReason

class nidcpower.**OutputCutoffReason**

ALL

Queries any output cutoff condition; clears all output cutoff conditions.

VOLTAGE_OUTPUT_HIGH

Queries or clears cutoff conditions when the output exceeded the high cutoff limit for voltage output.

VOLTAGE_OUTPUT_LOW

Queries or clears cutoff conditions when the output fell below the low cutoff limit for voltage output.

CURRENT_MEASURE_HIGH

Queries or clears cutoff conditions when the measured current exceeded the high cutoff limit for current output.

CURRENT_MEASURE_LOW

Queries or clears cutoff conditions when the measured current fell below the low cutoff limit for current output.

VOLTAGE_CHANGE_HIGH

Queries or clears cutoff conditions when the voltage slew rate increased beyond the positive change cutoff for voltage output.

VOLTAGE_CHANGE_LOW

Queries or clears cutoff conditions when the voltage slew rate decreased beyond the negative change cutoff for voltage output.

CURRENT_CHANGE_HIGH

Queries or clears cutoff conditions when the current slew rate increased beyond the positive change cutoff for current output.

CURRENT_CHANGE_LOW

Queries or clears cutoff conditions when the current slew rate decreased beyond the negative change cutoff for current output.

CURRENT_SATURATED

Queries or clears cutoff conditions when the measured current saturates the current range.

VOLTAGE_MEASURE_HIGH

Queries or clears cutoff conditions when the measured voltage exceeded the high cutoff limit for voltage output.

VOLTAGE_MEASURE_LOW

Queries or clears cutoff conditions when the measured voltage fell below the low cutoff limit for voltage output.

OutputFunction

`class nidcpower.OutputFunction`

DC_VOLTAGE

Sets the output method to DC voltage.

DC_CURRENT

Sets the output method to DC current.

PULSE_VOLTAGE

Sets the output method to pulse voltage.

PULSE_CURRENT

Sets the output method to pulse current.

OutputStates

`class nidcpower.OutputStates`

VOLTAGE

The channel maintains a constant voltage by adjusting the current.

CURRENT

The channel maintains a constant current by adjusting the voltage.

Polarity

`class nidcpower.Polarity`

HIGH

A high pulse occurs when the event is generated. The exported signal is low level both before and after the event is generated.

LOW

A low pulse occurs when the event is generated. The exported signal is high level both before and after the event is generated.

PowerAllocationMode

class nidcpower.**PowerAllocationMode**

DISABLED

The device attempts to source, on each active channel, the power that the present source configuration requires; NI-DCPower does not perform a sourcing power check. If the required power is greater than the maximum sourcing power, the device attempts to source the required amount and may shut down to prevent damage.

AUTOMATIC

The device attempts to source, on each active channel, the power that the present source configuration requires; NI-DCPower performs a sourcing power check. If the required power is greater than the maximum sourcing power, the device does not exceed the maximum power, and NI-DCPower returns an error.

MANUAL

The device attempts to source, on each active channel, the power you request with the *nidcpower.Session.requested_power_allocation* property; NI-DCPower performs a sourcing power check. If the requested power is either less than the required power for the present source configuration or greater than the maximum sourcing power, the device does not exceed the requested or allowed power, respectively, and NI-DCPower returns an error.

PowerSource

class nidcpower.**PowerSource**

INTERNAL

Uses the PXI chassis power source.

AUXILIARY

Uses the auxiliary power source connected to the device.

AUTOMATIC

Uses the auxiliary power source if it is available; otherwise uses the PXI chassis power source.

PowerSourceInUse

class nidcpower.**PowerSourceInUse**

INTERNAL

Uses the PXI chassis power source.

AUXILIARY

Uses the auxiliary power source connected to the device. Only the NI PXI-4110, NI PXIe-4112, NI PXIe-4113, and NI PXI-4130 support this value. This is the only supported value for the NI PXIe-4112 and NI PXIe-4113.

SelfCalibrationPersistence

class nidcpower.**SelfCalibrationPersistence**

KEEP_IN_MEMORY

Keep new self calibration values in memory only.

WRITE_TO_EEPROM

Write new self calibration values to hardware.

SendSoftwareEdgeTriggerType

class nidcpower.**SendSoftwareEdgeTriggerType**

START

Asserts the Start trigger.

SOURCE

Asserts the Source trigger.

MEASURE

Asserts the Measure trigger.

SEQUENCE_ADVANCE

Asserts the Sequence Advance trigger.

PULSE

Asserts the Pulse trigger.

SHUTDOWN

Asserts the Shutdown trigger.

Sense

class nidcpower.**Sense**

LOCAL

Local sensing is selected.

REMOTE

Remote sensing is selected.

SourceMode

class nidcpower.**SourceMode**

SINGLE_POINT

The source unit applies a single source configuration.

SEQUENCE

The source unit applies a list of voltage or current configurations sequentially.

TransientResponse

class nidcpower.**TransientResponse**

NORMAL

The output responds to changes in load at a normal speed.

FAST

The output responds to changes in load quickly.

SLOW

The output responds to changes in load slowly.

CUSTOM

The output responds to changes in load based on specified values.

TriggerType

class nidcpower.**TriggerType**

NONE

No trigger is configured.

DIGITAL_EDGE

The data operation starts when a digital edge is detected.

SOFTWARE_EDGE

The data operation starts when a software trigger occurs.

Exceptions and Warnings

Error

exception nidcpower.errors.**Error**

Base exception type that all NI-DCPower exceptions derive from

DriverError

exception nidcpower.errors.**DriverError**

An error originating from the NI-DCPower driver

UnsupportedConfigurationError

exception nidcpower.errors.**UnsupportedConfigurationError**

An error due to using this module in an usupported platform.

DriverNotInstalledError

exception nidcpower.errors.**DriverNotInstalledError**

An error due to using this module without the driver runtime installed.

DriverTooOldError

exception nidcpower.errors.**DriverTooOldError**

An error due to using this module with an older version of the NI-DCPower driver runtime.

DriverTooNewError

exception nidcpower.errors.**DriverTooNewError**

An error due to the NI-DCPower driver runtime being too new for this module.

InvalidRepeatedCapabilityError

exception nidcpower.errors.**InvalidRepeatedCapabilityError**

An error due to an invalid character in a repeated capability

SelfTestError

exception nidcpower.errors.**SelfTestError**

An error due to a failed self-test

RpcError

exception nidcpower.errors.**RpcError**

An error specific to sessions to the NI gRPC Device Server

DriverWarning

exception nidcpower.errors.**DriverWarning**

A warning originating from the NI-DCPower driver

Examples

You can download all nidcpower examples here

nidcpower_advanced_sequence.py

Listing 1: (nidcpower_advanced_sequence.py)

```
#!/usr/bin/python

import argparse
import hightime
import nidcpower
import sys


def example(resource_name, options, voltage_max, current_max, points_per_output_function, source_delay):
    with nidcpower.Session(resource_name=resource_name, options=options) as session:
        # Configure the session.
        session.source_mode = nidcpower.SourceMode.SEQUENCE
        session.voltage_level_autorange = True
        session.current_limit_autorange = True
        session.source_delay = hightime.timedelta(seconds=source_delay)
        properties_used = ['output_function', 'voltage_level', 'current_level']
        session.create_advanced_sequence(sequence_name='my_sequence', property_names=properties_used, set_as_active_sequence=True)

        voltage_per_step = voltage_max / points_per_output_function
        for i in range(points_per_output_function):
            session.create_advanced_sequence_step(set_as_active_step=False)
            session.output_function = nidcpower.OutputFunction.DC_VOLTAGE
            session.voltage_level = voltage_per_step * i

        current_per_step = current_max / points_per_output_function
        for i in range(points_per_output_function):
            session.create_advanced_sequence_step(set_as_active_step=False)
            session.output_function = nidcpower.OutputFunction.DC_CURRENT
            session.current_level = current_per_step * i

        # Calculate the timeout.
        aperture_time = session.aperture_time
        total_points = points_per_output_function * 2
        timeout = hightime.timedelta(seconds=((source_delay + aperture_time) * total_points + 1.0))

        with session.initiate():
            channel_indices = f'0-{session.channel_count - 1}'
            channels = session.get_channel_names(channel_indices)
            measurement_group = [session.channels[name].fetch_multiple(total_points, timeout=timeout) for name in channels]

        session.delete_advanced_sequence(sequence_name='my_sequence')
        line_format = '{:<15} {:<4} {:<10} {:<10} {:<6}'
        print(line_format.format('Channel', 'Num', 'Voltage', 'Current', 'In Compliance'))
        for i, measurements in enumerate(measurement_group):
```

(continues on next page)

(continued from previous page)

```
            num = 0
            channel_name = channels[i].strip()
            for measurement in measurements:
                print(line_format.format(channel_name, num, measurement.voltage,␣
↪measurement.current, str(measurement.in_compliance)))
                num += 1


def _main(argsv):
    parser = argparse.ArgumentParser(description='Output ramping voltage to voltage max,␣
↪then ramping current to current max.', formatter_class=argparse.
↪ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2/0, PXI1Slot3/0-1',␣
↪help='Resource names of NI SMUs.')
    parser.add_argument('-s', '--number-steps', default=256, type=int, help='Number of␣
↪steps per output function')
    parser.add_argument('-v', '--voltage-max', default=1.0, type=float, help='Maximum␣
↪voltage (V)')
    parser.add_argument('-i', '--current-max', default=0.001, type=float, help='Maximum␣
↪Current (I)')
    parser.add_argument('-d', '--delay', default=0.05, type=float, help='Source delay (s)
↪')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option␣
↪string')
    args = parser.parse_args(argsv)
    example(args.resource_name, args.option_string, args.voltage_max, args.current_max,␣
↪args.number_steps, args.delay)


def main():
    _main(sys.argv[1:])


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:4162; BoardType:PXIe',␣
↪]
    _main(cmd_line)


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '4162', 'BoardType': 'PXIe', }
↪, }
    example('PXI1Slot2/0, PXI1Slot3/1', options, 1.0, 0.001, 256, 0.05)


if __name__ == '__main__':
    main()


```

nidcpower_lcr_source_ac_voltage.py

Listing 2: (nidcpower_lcr_source_ac_voltage.py)

```
#!/usr/bin/python

import argparse
import nidcpower
import sys


def example(
    resource_name,
    options,
    lcr_frequency,
    lcr_impedance_range,
    cable_length,
    lcr_voltage_rms,
    lcr_dc_bias_source,
    lcr_dc_bias_voltage_level,
    lcr_measurement_time,
    lcr_custom_measurement_time,
    lcr_source_delay_mode,
    source_delay,
):
    with nidcpower.Session(resource_name=resource_name, options=options) as session:
        # Configure the session.
        session.instrument_mode = nidcpower.InstrumentMode.LCR
        session.lcr_stimulus_function = nidcpower.LCRStimulusFunction.VOLTAGE
        session.lcr_frequency = lcr_frequency
        session.lcr_impedance_range = lcr_impedance_range
        session.cable_length = cable_length
        session.lcr_voltage_amplitude = lcr_voltage_rms
        session.lcr_dc_bias_source = lcr_dc_bias_source
        session.lcr_dc_bias_voltage_level = lcr_dc_bias_voltage_level
        session.lcr_measurement_time = lcr_measurement_time
        session.lcr_custom_measurement_time = lcr_custom_measurement_time
        session.lcr_source_delay_mode = lcr_source_delay_mode
        session.source_delay = source_delay

        with session.initiate():
            # Low frequencies require longer settling times than the default timeout for
            # wait_for_event(), hence 5.0s is set here as a reasonable timeout value
            session.wait_for_event(event_id=nidcpower.Event.SOURCE_COMPLETE, timeout=5.0)
            measurements = session.measure_multiple_lcr()
            for measurement in measurements:
                print(measurement)

        session.reset()


def _main(argsv):
    parser = argparse.ArgumentParser(
```

(continues on next page)

(continued from previous page)

```
        description='Output the specified AC voltage and DC bias voltage, then takes LCR measurements',
        formatter_class=argparse.ArgumentDefaultsHelpFormatter
    )
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2/0', help='Resource names of NI SMUs')
    parser.add_argument('-f', '--lcr-frequency', default=10.0e3, type=float, help='LCR frequency (Hz)')
    parser.add_argument('-i', '--lcr-impedance-range', default=100.0, type=float, help='LCR impedance range ()')
    parser.add_argument('-c', '--cable-length', default='NI_STANDARD_2M', type=str, choices=tuple(nidcpower.CableLength.__members__.keys()), help='Cable length')
    parser.add_argument('-v', '--lcr-voltage-rms', default=700.0e-3, type=float, help='LCR voltage RMS (V RMS)')
    parser.add_argument('-d', '--lcr-dc-bias-source', default='OFF', type=str, choices=tuple(nidcpower.LCRDCBiasSource.__members__.keys()), help='LCR DC bias source')
    parser.add_argument('-dv', '--lcr-dc-bias-voltage_level', default=0.0, type=float, help='LCR DC bias voltage (V)')
    parser.add_argument('-t', '--lcr-measurement-time', default='MEDIUM', type=str, choices=tuple(nidcpower.LCRMeasurementTime.__members__.keys()), help='LCR measurement time')
    parser.add_argument('-ct', '--lcr-custom-measurement-time', default=10.0e-3, type=float, help='LCR custom measurement time (s)')
    parser.add_argument('-sm', '--lcr-source-delay-mode', default='AUTOMATIC', type=str, choices=tuple(nidcpower.LCRSourceDelayMode.__members__.keys()), help='LCR source delay mode')
    parser.add_argument('-s', '--source-delay', default=16.66e-3, type=float, help='Source delay (s)')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option string')
    args = parser.parse_args(argsv)
    example(
        resource_name=args.resource_name,
        options=args.option_string,
        lcr_frequency=args.lcr_frequency,
        lcr_impedance_range=args.lcr_impedance_range,
        cable_length=getattr(nidcpower.CableLength, args.cable_length),
        lcr_voltage_rms=args.lcr_voltage_rms,
        lcr_dc_bias_source=getattr(nidcpower.LCRDCBiasSource, args.lcr_dc_bias_source),
        lcr_dc_bias_voltage_level=args.lcr_dc_bias_voltage_level,
        lcr_measurement_time=getattr(nidcpower.LCRMeasurementTime, args.lcr_measurement_time),
        lcr_custom_measurement_time=args.lcr_custom_measurement_time,
        lcr_source_delay_mode=getattr(nidcpower.LCRSourceDelayMode, args.lcr_source_delay_mode),
        source_delay=args.source_delay,
    )


def main():
    _main(sys.argv[1:])

```

(continues on next page)

(continued from previous page)

```

def test_example():
    example(
        resource_name='PXI1Slot2/0',
        options={'simulate': True, 'driver_setup': {'Model': '4190', 'BoardType': 'PXIe',
↪ }, },
        lcr_frequency=10.0e3,
        lcr_impedance_range=100.0,
        cable_length=nidcpower.CableLength.NI_STANDARD_2M,
        lcr_voltage_rms=700.0e-3,
        lcr_dc_bias_source=nidcpower.LCRDCBiasSource.OFF,
        lcr_dc_bias_voltage_level=0.0,
        lcr_measurement_time=nidcpower.LCRMeasurementTime.MEDIUM,
        lcr_custom_measurement_time=10.0e-3,
        lcr_source_delay_mode=nidcpower.LCRSourceDelayMode.AUTOMATIC,
        source_delay=16.66e-3,
    )


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:4190; BoardType:PXIe',␣
↪]
    _main(cmd_line)


if __name__ == '__main__':
    main()
```

nidcpower_measure_record.py

Listing 3: (nidcpower_measure_record.py)

```
#!/usr/bin/python

import argparse
import nidcpower
import sys


def example(resource_name, options, voltage, length):
    with nidcpower.Session(resource_name=resource_name, options=options) as session:
        # Configure the session.
        session.measure_record_length = length
        session.measure_record_length_is_finite = True
        session.measure_when = nidcpower.MeasureWhen.AUTOMATICALLY_AFTER_SOURCE_COMPLETE
        session.output_function = nidcpower.OutputFunction.DC_VOLTAGE
        session.voltage_level = voltage

        session.commit()
        print(f'Effective measurement rate: {session.measure_record_delta_time / 1} S/s')
```

(continues on next page)

(continued from previous page)

```
        print('Channel           Num  Voltage    Current    In Compliance')
        row_format = '{0:15} {1:3d}    {2:8.6f}   {3:8.6f}   {4}'
        with session.initiate():
            channel_indices = f'0-{session.channel_count - 1}'
            channels = session.get_channel_names(channel_indices)
            for i, channel_name in enumerate(channels):
                samples_acquired = 0
                while samples_acquired < length:
                    measurements = session.channels[channel_name].fetch_
↪multiple(count=session.fetch_backlog)
                    samples_acquired += len(measurements)
                    for i in range(len(measurements)):
                        print(row_format.format(channel_name, i, measurements[i].voltage,
↪ measurements[i].current, measurements[i].in_compliance))


def _main(argsv):
    parser = argparse.ArgumentParser(description='Outputs the specified voltage, then␣
↪takes the specified number of voltage and current readings.', formatter_class=argparse.
↪ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2/0, PXI1Slot3/0-1',␣
↪help='Resource names of NI SMUs.')
    parser.add_argument('-l', '--length', default='20', type=int, help='Measure record␣
↪length per channel')
    parser.add_argument('-v', '--voltage', default=5.0, type=float, help='Voltage level␣
↪(V)')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option␣
↪string')
    args = parser.parse_args(argsv)
    example(args.resource_name, args.option_string, args.voltage, args.length)


def main():
    _main(sys.argv[1:])


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '4162', 'BoardType': 'PXIe', }
↪, }
    example('PXI1Slot2/0, PXI1Slot3/1', options, 5.0, 20)


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:4162; BoardType:PXIe',␣
↪]
    _main(cmd_line)


if __name__ == '__main__':
    main()
```

nidcpower_source_delay_measure.py

Listing 4: (nidcpower_source_delay_measure.py)

```
#!/usr/bin/python

import argparse
import hightime
import nidcpower
import sys


def print_fetched_measurements(measurements):
    print(f'             Voltage : {measurements[0].voltage:f} V')
    print(f'              Current: {measurements[0].current:f} A')
    print(f'        In compliance: {measurements[0].in_compliance}')


def example(resource_name, options, voltage1, voltage2, delay):
    timeout = hightime.timedelta(seconds=(delay + 1.0))

    with nidcpower.Session(resource_name=resource_name, options=options) as session:
        # Configure the session.
        session.source_mode = nidcpower.SourceMode.SINGLE_POINT
        session.output_function = nidcpower.OutputFunction.DC_VOLTAGE
        session.current_limit = .06
        session.voltage_level_range = 5.0
        session.current_limit_range = .06
        session.source_delay = hightime.timedelta(seconds=delay)
        session.measure_when = nidcpower.MeasureWhen.AUTOMATICALLY_AFTER_SOURCE_COMPLETE
        session.voltage_level = voltage1

        with session.initiate():
            channel_indices = f'0-{session.channel_count - 1}'
            channels = session.get_channel_names(channel_indices)
            for channel_name in channels:
                print(f'Channel: {channel_name}')
                print('---------------------------------')
                print('Voltage 1:')
                print_fetched_measurements(session.channels[channel_name].fetch_
↪multiple(count=1, timeout=timeout))
                session.voltage_level = voltage2  # on-the-fly set
                print('Voltage 2:')
                print_fetched_measurements(session.channels[channel_name].fetch_
↪multiple(count=1, timeout=timeout))
                session.output_enabled = False
                print('')


def _main(argsv):
    parser = argparse.ArgumentParser(description='Outputs voltage 1, waits for source␣
↪delay, and then takes a measurement. Then orepeat with voltage 2.', formatter_
↪class=argparse.ArgumentDefaultsHelpFormatter)
```

(continues on next page)

(continued from previous page)

```
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2/0, PXI1Slot3/0-1',␣
↪help='Resource names of an NI SMUs.')
    parser.add_argument('-v1', '--voltage1', default=1.0, type=float, help='Voltage␣
↪level 1 (V)')
    parser.add_argument('-v2', '--voltage2', default=2.0, type=float, help='Voltage␣
↪level 2 (V)')
    parser.add_argument('-d', '--delay', default=0.05, type=float, help='Source delay (s)
↪')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option␣
↪string')
    args = parser.parse_args(argsv)
    example(args.resource_name, args.option_string, args.voltage1, args.voltage2, args.
↪delay)


def main():
    _main(sys.argv[1:])


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:4162; BoardType:PXIe',␣
↪]
    _main(cmd_line)


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '4162', 'BoardType': 'PXIe', }
↪, }
    example('PXI1Slot2/0, PXI1Slot3/1', options, 1.0, 2.0, 0.05)


if __name__ == '__main__':
    main()


```

gRPC Support

Support for using NI-DCPower over gRPC

SessionInitializationBehavior

class nidcpower.**SessionInitializationBehavior**

AUTO

The NI gRPC Device Server will attach to an existing session with the specified name if it exists, otherwise the server will initialize a new session.

Note: When using the Session as a context manager and the context exits, the behavior depends on what happened when the constructor was called. If it resulted in a new session being initialized on the NI gRPC

Device Server, then it will automatically close the server session. If it instead attached to an existing session, then it will detach from the server session and leave it open.

INITIALIZE_SERVER_SESSION

Require the NI gRPC Device Server to initialize a new session with the specified name.

Note: When using the Session as a context manager and the context exits, it will automatically close the server session.

ATTACH_TO_SERVER_SESSION

Require the NI gRPC Device Server to attach to an existing session with the specified name.

Note: When using the Session as a context manager and the context exits, it will detach from the server session and leave it open.

GrpcSessionOptions

class nidcpower.**GrpcSessionOptions**(*self*, *grpc_channel*, *session_name*, *initialization_behavior=SessionInitializationBehavior.AUTO*)

Collection of options that specifies session behaviors related to gRPC.

Creates and returns an object you can pass to a Session constructor.

Parameters

- **grpc_channel** (*grpc.Channel*) – Specifies the channel to the NI gRPC Device Server.
- **session_name** (*str*) – User-specified name that identifies the driver session on the NI gRPC Device Server.

  This is different from the resource name parameter many APIs take as a separate parameter. Specifying a name makes it easy to share sessions across multiple gRPC clients. You can use an empty string if you want to always initialize a new session on the server. To attach to an existing session, you must specify the session name it was initialized with.
- **initialization_behavior** (*nidcpower.SessionInitializationBehavior*) – Specifies whether it is acceptable to initialize a new session or attach to an existing one, or if only one of the behaviors is desired.

  The driver session exists on the NI gRPC Device Server.

4.2 Additional Documentation

Refer to your driver documentation for device-specific information and detailed API documentation.

Refer to the nimi-python Read the Docs project for documentation of versions 1.4.4 of the module or earlier.

CHAPTER

FIVE

LICENSE

nimi-python is licensed under an MIT-style license (see LICENSE). Other incorporated projects may be licensed under different licenses. All licenses allow for non-commercial and commercial use.

gRPC Features

For driver APIs that support it, passing a GrpcSessionOptions instance as a parameter to Session.__init__() is subject to the NI General Purpose EULA (see NILICENSE).

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

A

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z