NI-DMM Python API Documentation

Release 1.4.9.dev0

NI

May 01, 2024

DOCUMENTATION

1 About **1**
- 1.1 Support Policy . . . 1

2 Contributing **3**

3 Support / Feedback **5**

4 Bugs / Feature Requests **7**
- 4.1 nidmm module . . . 7
  - 4.1.1 Installation . . . 7
  - 4.1.2 Usage . . . 7
  - 4.1.3 API Reference . . . 7
- 4.2 Additional Documentation . . . 79

5 License **81**

6 Indices and tables **83**

Python Module Index **85**

Index **87**

CHAPTER

ONE

ABOUT

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

1.1 Support Policy

nidmm supports all the Operating Systems supported by NI-DMM.

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 nidmm module

4.1.1 Installation

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

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

```
$ python -m pip install nidmm
```

4.1.2 Usage

The following is a basic example of using the **nidmm** module to open a session to a DMM and perform a 5.5 digits of resolution voltage measurement in the 10 V range.

```
import nidmm
with nidmm.Session("Dev1") as session:
    session.configure_measurement_digits(nidmm.Function.DC_VOLTS, 10.0, 5.5)
    print("Measurement: " + str(session.read()))
```

Other usage examples can be found on GitHub.

4.1.3 API Reference

Session

class nidmm.**Session**(*self*, *resource_name*, *id_query=False*, *reset_device=False*, *options={}*, *, *grpc_options=None*)

This method completes the following tasks:

- Creates a new IVI instrument driver session and, optionally, sets the initial state of the following session properties: `nidmm.Session.RANGE_CHECK`, `nidmm.Session.QUERY_INSTR_STATUS`, `nidmm.Session.CACHE`, *nidmm.Session.simulate*, `nidmm.Session.RECORD_COERCIONS`.

- Opens a session to the device you specify for the **Resource_Name** parameter. If the **ID_Query** parameter is set to True, this method queries the instrument ID and checks that it is valid for this instrument driver.
- If the **Reset_Device** parameter is set to True, this method resets the instrument to a known state. Sends initialization commands to set the instrument to the state necessary for the operation of the instrument driver.
- Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver method calls.

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

Parameters

- **resource_name** (*str*) –

  Caution: All IVI names for the **Resource_Name**, such as logical names or virtual names, are case-sensitive. If you use logical names, driver session names, or virtual names in your program, you must make sure that the name you use matches the name in the IVI Configuration Store file exactly, without any variations in the case of the characters in the name.

  Contains the **resource_name** of the device to initialize. The **resource_name** is assigned in Measurement & Automation Explorer (MAX). Refer to Related Documentation for the *NI Digital Multimeters Getting Started Guide* for more information about configuring and testing the DMM in MAX.
  Valid Syntax:

  - NI-DAQmx name
  - DAQ::NI-DAQmx name[::INSTR]
  - DAQ::Traditional NI-DAQ device number[::INSTR]
  - IVI logical name

- **id_query** (*bool*) – Verifies that the device you initialize is one that the driver supports. NI-DMM automatically performs this query, so setting this parameter is not necessary. Defined Values:

True (default)	1	Perform ID Query
False	0	Skip ID Query

- **reset_device** (*bool*) – Specifies whether to reset the instrument during the initialization procedure. Defined Values:

True (default)	1	Reset Device
False	0	Don't Reset

- **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	{}

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

Methods

abort

nidmm.Session.**abort**()

Aborts a previously initiated measurement and returns the DMM to the Idle state.

close

nidmm.Session.**close**()

Closes the specified session and deallocates resources that it reserved.

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

configure_measurement_absolute

nidmm.Session.**configure_measurement_absolute**(*measurement_function*, *range*, *resolution_absolute*)

Configures the common properties of the measurement. These properties include nidmm.Session.method, *nidmm.Session.range*, and *nidmm.Session.resolution_absolute*.

Parameters

- **measurement_function** (*nidmm.Function*) – Specifies the **measurement_function** used to acquire the measurement. The driver sets nidmm.Session.method to this value.
- **range** (*float*) – Specifies the **range** for the method specified in the **Measurement_Function** parameter. When frequency is specified in the **Measurement_Function** parameter, you must supply the minimum frequency expected in the **range** parameter. For example, you must type in 100 Hz if you are measuring

101 Hz or higher. For all other methods, you must supply a **range** that exceeds the value that you are measuring. For example, you must type in 10 V if you are measuring 9 V. **range** values are coerced up to the closest input **range**. Refer to the Devices Overview for a list of valid ranges. The driver sets *nidmm.Session.range* to this value. The default is 0.02 V.

NIDMM_VAL	-1.0	NI-DMM performs an Auto Range before acquiring the measurement.
NIDMM_VAL	-2.0	NI-DMM sets the Range to the current *nidmm.Session.auto_range_value* and uses this range for all subsequent measurements until the measurement configuration is changed.
NIDMM_VAL	-3.0	NI-DMM performs an Auto Range before acquiring the measurement. The *nidmm.Session.auto_range_value* is stored and used for all subsequent measurements until the measurement configuration is changed.

Note: The NI 4050, NI 4060, and NI 4065 only support Auto Range when the trigger and sample trigger are set to IMMEDIATE.

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.

- **resolution_absolute** (*float*) – Specifies the absolute resolution for the measurement. NI-DMM sets *nidmm.Session.resolution_absolute* to this value. The PXIe-4080/4081/4082 uses the resolution you specify. The NI 4065 and NI 4070/4071/4072 ignore this parameter when the **Range** parameter is set to NIDMM_VAL_AUTO_RANGE_ON (-1.0) or NIDMM_VAL_AUTO_RANGE_ONCE (-3.0). The default is 0.001 V.

  Note: NI-DMM ignores this parameter for capacitance and inductance measurements on the NI 4072. To achieve better resolution for such measurements, use the *nidmm.Session.lc_number_meas_to_average* property.

  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.

configure_measurement_digits

nidmm.Session.**configure_measurement_digits**(*measurement_function*, *range*, *resolution_digits*)

Configures the common properties of the measurement. These properties include `nidmm.Session.method`, *nidmm.Session.range*, and *nidmm.Session.resolution_digits*.

Parameters

- **measurement_function** (*nidmm.Function*) – Specifies the **measurement_function** used to acquire the measurement. The driver sets `nidmm.Session.method` to this value.
- **range** (*float*) – Specifies the range for the method specified in the **Measurement_Function** parameter. When frequency is specified in the **Measurement_Function** parameter, you must supply the minimum frequency expected in the **range** parameter. For example, you must type in 100 Hz if you are measuring 101 Hz or higher. For all other methods, you must supply a range that exceeds the value that you are measuring. For example, you must type in 10 V if you are measuring 9 V. range values are coerced up to the closest input range. Refer to the Devices Overview for a list of valid ranges. The driver sets *nidmm.Session.range* to this value. The default is 0.02 V.

NIDMM_VA	-1.0	NI-DMM performs an Auto Range before acquiring the measurement.
NIDMM_VA	-2.0	NI-DMM sets the Range to the current *nidmm.Session.auto_range_value* and uses this range for all subsequent measurements until the measurement configuration is changed.
NIDMM_VA	-3.0	NI-DMM performs an Auto Range before acquiring the measurement. The *nidmm.Session.auto_range_value* is stored and used for all subsequent measurements until the measurement configuration is changed.

Note: The NI 4050, NI 4060, and NI 4065 only support Auto Range when the trigger and sample trigger are set to IMMEDIATE.

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.

- **resolution_digits** (*float*) – Specifies the resolution of the measurement in digits. The driver sets the Devices Overview for a list of valid ranges. The driver sets *nidmm.Session.resolution_digits* property to this value. The PXIe-4080/4081/4082 uses the resolution you specify. The NI 4065 and NI 4070/4071/4072 ignore this parameter when the **Range** parameter is set to `NIDMM_VAL_AUTO_RANGE_ON` (-1.0) or `NIDMM_VAL_AUTO_RANGE_ONCE` (-3.0). The default is 5½.

Note: NI-DMM ignores this parameter for capacitance and inductance measurements on the NI 4072. To achieve better resolution for such measurements, use the

nidmm.Session.lc_number_meas_to_average property.

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.

configure_multi_point

nidmm.Session.**configure_multi_point**(*trigger_count*, *sample_count*, *sample_trigger=nidmm.SampleTrigger.IMMEDIATE*, *sample_interval=hightime.timedelta(seconds=-1)*)

Configures the properties for multipoint measurements. These properties include *nidmm.Session.trigger_count*, *nidmm.Session.sample_count*, *nidmm.Session.sample_trigger*, and *nidmm.Session.sample_interval*.

For continuous acquisitions, set *nidmm.Session.trigger_count* or *nidmm.Session.sample_count* to zero. For more information, refer to Multiple Point Acquisitions, Triggering, and Using Switches.

Parameters

- **trigger_count** (*int*) – Sets the number of triggers you want the DMM to receive before returning to the Idle state. The driver sets *nidmm.Session.trigger_count* to this value. The default value is 1.
- **sample_count** (*int*) – Sets the number of measurements the DMM makes in each measurement sequence initiated by a trigger. The driver sets *nidmm.Session.sample_count* to this value. The default value is 1.
- **sample_trigger** (*nidmm.SampleTrigger*) – Specifies the **sample_trigger** source you want to use. The driver sets *nidmm.Session.sample_trigger* to this value. The default is Immediate.

  Note: To determine which values are supported by each device, refer to the Lab-Windows/CVI Trigger Routing section.

- **sample_interval** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Sets the amount of time in seconds the DMM waits between measurement cycles. The driver sets *nidmm.Session.sample_interval* to this value. Specify a sample interval to add settling time between measurement cycles or to decrease the measurement rate. **sample_interval** only applies when the **Sample_Trigger** is set to INTERVAL.

  On the NI 4060, the **sample_interval** value is used as the settling time. When sample interval is set to 0, the DMM does not settle between measurement cycles. The NI 4065 and NI 4070/4071/4072 use the value specified in **sample_interval** as additional delay. The default value (-1) ensures that the DMM settles for a recommended time. This is the same as using an Immediate trigger.

  Note: This property is not used on the NI 4080/4081/4082 and the NI 4050.

configure_rtd_custom

nidmm.Session.**configure_rtd_custom**(*rtd_a*, *rtd_b*, *rtd_c*)

Configures the A, B, and C parameters for a custom RTD.

Parameters

- **rtd_a** (*float*) – Specifies the Callendar-Van Dusen A coefficient for RTD scaling when RTD Type parameter is set to Custom in the *nidmm.Session.configure_rtd_type()* method. The default is 3.9083e-3 (Pt3851)
- **rtd_b** (*float*) – Specifies the Callendar-Van Dusen B coefficient for RTD scaling when RTD Type parameter is set to Custom in the *nidmm.Session.configure_rtd_type()* method. The default is -5.775e-7 (Pt3851).
- **rtd_c** (*float*) – Specifies the Callendar-Van Dusen C coefficient for RTD scaling when RTD Type parameter is set to Custom in the *nidmm.Session.configure_rtd_type()* method. The default is -4.183e-12 (Pt3851).

configure_rtd_type

nidmm.Session.**configure_rtd_type**(*rtd_type*, *rtd_resistance*)

Configures the RTD Type and RTD Resistance parameters for an RTD.

Parameters

- **rtd_type** (*nidmm.RTDType*) – Specifies the type of RTD used to measure the temperature resistance. NI-DMM uses this value to set the RTD Type property. The default is *PT3851*.

Enum	Standards	Material	TCR ()	Typical R_0 ()	Notes	
Callendar-Van Dusen Coefficient						
PT3851	IEC-751 DIN 43760 BS 1904 ASTM-E1137 EN-60751	Platinum	.0038	100 1000	A = 3.9083 × 10^{-3} B = –5.775×10:sup:–*7* C = –4.183×10:sup:–*12*	Most common RTDs
PT3750	Low-cost vendor compliant RTD*	Platinum	.0037	1000	A = 3.81 × 10^{-3} B = –6.02×10:sup:–*7* C = –6.0×10:sup:–*12*	Low-cost RTD
PT3916	JISC 1604	Platinum	.0039	100	A = 3.9739 × 10^{-3} B = –5.870×10:sup:–*7* C = –4.4 × 10^{-12}	Used in primarily in Japan
PT3920	US Industrial Standard D-100 American	Platinum	.0039	100	A = 3.9787 × 10^{-3} B = –5.8686×10:sup:–*7* C = –4.167 × 10^{-12}	Low-cost RTD
PT3911	US Industrial Standard American	Platinum	.0039	100	A = 3.9692 × 10^{-3} B = –5.8495×10:sup:–*7* C = –4.233 × 10^{-12}	Low-cost RTD
PT3928	ITS-90	Platinum	.0039	100	A = 3.9888 × 10^{-3} B = –5.915×10:sup:–*7* C = –3.85 × 10^{-12}	The definition of temperature
*No standard. Check the TCR.						

- **rtd_resistance** (*float*) – Specifies the RTD resistance in ohms at 0 °C. NI-DMM uses this value to set the RTD Resistance property. The default is 100 (Ω).

configure_thermistor_custom

nidmm.Session.**configure_thermistor_custom**(*thermistor_a*, *thermistor_b*, *thermistor_c*)

Configures the A, B, and C parameters for a custom thermistor.

Parameters

- **thermistor_a** (*float*) – Specifies the Steinhart-Hart A coefficient for thermistor scaling when Thermistor Type is set to Custom in the nidmm.Session.ConfigureThermistorType() method. The default is 1.0295e-3 (44006).

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

- **thermistor_b** (*float*) – Specifies the Steinhart-Hart B coefficient for thermistor scaling when Thermistor Type is set to Custom in the nidmm.Session.ConfigureThermistorType() method. The default is 2.391e-4 (44006).

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

- **thermistor_c** (*float*) – Specifies the Steinhart-Hart C coefficient for thermistor scaling when Thermistor Type is set to Custom in the nidmm.Session.ConfigureThermistorType() method. The default is 1.568e-7 (44006).

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

configure_thermocouple

nidmm.Session.**configure_thermocouple**(*thermocouple_type*, *reference_junction_type=nidmm.ThermocoupleReferenceJunctionType.FIXED*)

Configures the thermocouple type and reference junction type for a chosen thermocouple.

Parameters

- **thermocouple_type** (*nidmm.ThermocoupleType*) – Specifies the type of thermocouple used to measure the temperature. NI-DMM uses this value to set the Thermocouple Type property. The default is *J*.

B	Thermocouple type B
E	Thermocouple type E
J	Thermocouple type J
K	Thermocouple type K
N	Thermocouple type N
R	Thermocouple type R
S	Thermocouple type S
T	Thermocouple type T

- **reference_junction_type** (*nidmm.ThermocoupleReferenceJunctionType*) – Specifies the type of reference junction to be used in the reference junction compensation of a thermocouple measurement. NI-DMM uses this value to set the Reference Junction Type property. The only supported value is *FIXED*.

configure_trigger

nidmm.Session.**configure_trigger**(*trigger_source*, *trigger_delay=hightime.timedelta(seconds=-1)*)

Configures the DMM **Trigger_Source** and **Trigger_Delay**. Refer to Triggering and Using Switches for more information.

Parameters

- **trigger_source** (*nidmm.TriggerSource*) – Specifies the **trigger_source** that initiates the acquisition. The driver sets *nidmm.Session.trigger_source* to this value. Software configures the DMM to wait until *nidmm.Session.send_software_trigger()* is called before triggering the DMM.

  Note: To determine which values are supported by each device, refer to the Lab-Windows/CVI Trigger Routing section.

- **trigger_delay** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Specifies the time that the DMM waits after it has received a trigger before taking a measurement. The driver sets the *nidmm.Session.trigger_delay* property to this value. By default, **trigger_delay** is NIDMM_VAL_AUTO_DELAY (-1), which means the DMM waits an appropriate settling time before taking the measurement. On the NI 4060, if you set **trigger_delay** to 0, the DMM does not settle before taking the measurement. The NI 4065 and NI 4070/4071/4072 use the value specified in **trigger_delay** as additional settling time.

  Note: When using the NI 4050, **Trigger_Delay** must be set to NIDMM_VAL_AUTO_DELAY (-1).

  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.

configure_waveform_acquisition

nidmm.Session.**configure_waveform_acquisition**(*measurement_function*, *range*, *rate*, *waveform_points*)

Configures the DMM for waveform acquisitions. This feature is supported on the NI 4080/4081/4082 and the NI 4070/4071/4072.

Parameters

- **measurement_function** (*nidmm.Function*) – Specifies the **measurement_function** used in a waveform acquisition. The driver sets nidmm.Session.method to this value.

WAVEFORM_VOLTAGE (default)	1003	Voltage Waveform
WAVEFORM_CURRENT	1004	Current Waveform

- **range** (*float*) – Specifies the expected maximum amplitude of the input signal and sets the **range** for the **Measurement_Function**. NI-DMM sets *nidmm.Session.range* to this value. **range** values are coerced up to the closest input **range**. The default is 10.0.

  For valid ranges refer to the topics in Devices.

  Auto-ranging is not supported during waveform acquisitions.

- **rate** (*float*) – Specifies the **rate** of the acquisition in samples per second. NI-DMM sets *nidmm.Session.waveform_rate* to this value.

  The valid **Range** is 10.0–1,800,000 S/s. **rate** values are coerced to the closest integer divisor of 1,800,000. The default value is 1,800,000.

- **waveform_points** (*int*) – Specifies the number of points to acquire before the waveform acquisition completes. NI-DMM sets *nidmm.Session.waveform_points* to this value.

  To calculate the maximum and minimum number of waveform points that you can acquire in one acquisition, refer to the Waveform Acquisition Measurement Cycle.

  The default value is 500.

disable

nidmm.Session.**disable**()

Places the instrument in a quiescent state where it has minimal or no impact on the system to which it is connected. If a measurement is in progress when this method is called, the measurement is aborted.

export_attribute_configuration_buffer

nidmm.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.

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

Coercion Behavior for Certain Devices

Imported and exported property configurations contain coerced values for the following NI-DMM devices:

- PXI/PCI/PCIe/USB-4065
- PXI/PCI-4070
- PXI-4071
- PXI-4072

NI-DMM coerces property values when the value you set is within the allowed range for the property but is not one of the discrete valid values the property supports. For example, for a property that coerces values up, if you choose a value of 4 when the adjacent valid values are 1 and 10, the property coerces the value to 10.

Related Topics:

Using Properties and Properties with NI-DMM

Setting Properties Before Reading Properties

Note: Not supported on the PCMCIA-4050 or the PXI/PCI-4060.

Return type
bytes

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

export_attribute_configuration_file

nidmm.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.

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

Coercion Behavior for Certain Devices

Imported and exported property configurations contain coerced values for the following NI-DMM devices:

- PXI/PCI/PCIe/USB-4065
- PXI/PCI-4070
- PXI-4071
- PXI-4072

NI-DMM coerces property values when the value you set is within the allowed range for the property but is not one of the discrete valid values the property supports. For example, for a property that coerces values up, if you choose a value of 4 when the adjacent valid values are 1 and 10, the property coerces the value to 10.

Related Topics:

Using Properties and Properties with NI-DMM

Setting Properties Before Reading Properties

Note: Not supported on the PCMCIA-4050 or the PXI/PCI-4060.

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:** .nidmmconfig

fetch

nidmm.Session.**fetch**(*maximum_time=hightime.timedelta(milliseconds=-1)*)

Returns the value from a previously initiated measurement. You must call nidmm.Session._initiate() before calling this method.

Parameters

maximum_time (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the **maximum_time** allowed for this method to complete in milliseconds. If the method does not complete within this time interval, the method returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.

The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.

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.

Return type

float

Returns

The measured value returned from the DMM.

fetch_multi_point

nidmm.Session.**fetch_multi_point**(*array_size*, *maximum_time=hightime.timedelta(milliseconds=-1)*)

Returns an array of values from a previously initiated multipoint measurement. The number of measurements the DMM makes is determined by the values you specify for the **Trigger_Count** and **Sample_Count** parameters of *nidmm.Session.configure_multi_point()*. You must first call nidmm.Session._initiate() to initiate a measurement before calling this method.

Parameters

- **array_size** (*int*) – Specifies the number of measurements to acquire. The maximum number of measurements for a finite acquisition is the (**Trigger Count** x **Sample Count**) parameters in *nidmm.Session.configure_multi_point()*.

  For continuous acquisitions, up to 100,000 points can be returned at once. The number of measurements can be a subset. The valid range is any positive ViInt32. The default value is 1.

- **maximum_time** (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the **maximum_time** allowed for this method to complete in milliseconds. If the method does not complete within this time interval, the method returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code.

This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.

The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.

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.

Return type
array.array("d")

Returns

An array of measurement values.

Note: The size of the **Reading_Array** must be at least the size that you specify for the **Array_Size** parameter.

---

fetch_waveform

nidmm.Session.**fetch_waveform**(*array_size*, *maximum_time=hightime.timedelta(milliseconds=-1)*)

For the NI 4080/4081/4082 and the NI 4070/4071/4072, returns an array of values from a previously initiated waveform acquisition. You must call nidmm.Session._initiate() before calling this method.

Parameters

- **array_size** (*int*) – Specifies the number of waveform points to return. You specify the total number of points that the DMM acquires in the **Waveform Points** parameter of *nidmm.Session.configure_waveform_acquisition()*. The default value is 1.
- **maximum_time** (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the **maximum_time** allowed for this method to complete in milliseconds. If the method does not complete within this time interval, the method returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.

  The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.

  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.

Return type
array.array("d")

Returns
Waveform Array is an array of measurement values stored in waveform data type.

fetch_waveform_into

nidmm.Session.**fetch_waveform_into**(*array_size*, *maximum_time=hightime.timedelta(milliseconds=-1)*)

For the NI 4080/4081/4082 and the NI 4070/4071/4072, returns an array of values from a previously initiated waveform acquisition. You must call `nidmm.Session._initiate()` before calling this method.

Parameters

- **waveform_array** (*numpy.array(dtype=numpy.float64)*) – **Waveform Array** is an array of measurement values stored in waveform data type.
- **maximum_time** (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the **maximum_time** allowed for this method to complete in milliseconds. If the method does not complete within this time interval, the method returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.

  The valid range is 0–86400000. The default value is `NIDMM_VAL_TIME_LIMIT_AUTO` (-1). The DMM calculates the timeout automatically.

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.

get_cal_date_and_time

nidmm.Session.**get_cal_date_and_time**(*cal_type*)

Returns the date and time of the last calibration performed.

Note: The NI 4050 and NI 4060 are not supported.

Parameters

cal_type (*int*) – Specifies the type of calibration performed (external or self-calibration).

`NIDMM_VAL_INTERNAL_AREA` (default)	0	Self-Calibration
`NIDMM_VAL_EXTERNAL_AREA`	1	External Calibration

Note: The NI 4065 does not support self-calibration.

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.

Return type
hightime.datetime

Returns
Indicates date and time of the last calibration.

get_dev_temp

nidmm.Session.**get_dev_temp**(*options=''*)

Returns the current **Temperature** of the device.

Note: The NI 4050 and NI 4060 are not supported.

Parameters
options (*str*) – Reserved.

Return type
float

Returns
Returns the current **temperature** of the device.

get_ext_cal_recommended_interval

nidmm.Session.**get_ext_cal_recommended_interval**()

Returns the recommended interval between external recalibration in **Months**.

Note: The NI 4050 and NI 4060 are not supported.

Return type
hightime.timedelta

Returns
Returns the recommended number of **months** between external calibrations.

get_last_cal_temp

nidmm.Session.**get_last_cal_temp**(*cal_type*)

Returns the **Temperature** during the last calibration procedure.

Note: The NI 4050 and NI 4060 are not supported.

Parameters
cal_type (*int*) – Specifies the type of calibration performed (external or self-calibration).

NIDMM_VAL_INTERNAL_AREA (default)	0	Self-Calibration
NIDMM_VAL_EXTERNAL_AREA	1	External Calibration

Note: The NI 4065 does not support self-calibration.

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.

Return type
float

Returns
Returns the **temperature** during the last calibration.

get_self_cal_supported

nidmm.Session.**get_self_cal_supported**()

Returns a Boolean value that expresses whether or not the DMM that you are using can perform self-calibration.

Return type
bool

Returns

Returns whether Self Cal is supported for the device specified by the given session.

True	1	The DMM that you are using can perform self-calibration.
False	0	The DMM that you are using cannot perform self-calibration.

import_attribute_configuration_buffer

nidmm.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.

Coercion Behavior for Certain Devices

Imported and exported property configurations contain coerced values for the following NI-DMM devices:

- PXI/PCI/PCIe/USB-4065
- PXI/PCI-4070
- PXI-4071
- PXI-4072

NI-DMM coerces property values when the value you set is within the allowed range for the property but is not one of the discrete valid values the property supports. For example, for a property that coerces values up, if you choose a value of 4 when the adjacent valid values are 1 and 10, the property coerces the value to 10.

Related Topics:

Using Properties and Properties with NI-DMM

Setting Properties Before Reading Properties

Note: Not supported on the PCMCIA-4050 or the PXI/PCI-4060.

Parameters

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

import_attribute_configuration_file

nidmm.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.

Coercion Behavior for Certain Devices

Imported and exported property configurations contain coerced values for the following NI-DMM devices:

- PXI/PCI/PCIe/USB-4065
- PXI/PCI-4070
- PXI-4071
- PXI-4072

NI-DMM coerces property values when the value you set is within the allowed range for the property but is not one of the discrete valid values the property supports. For example, for a property that coerces values up, if you choose a value of 4 when the adjacent valid values are 1 and 10, the property coerces the value to 10.

Related Topics:

Using Properties and Properties with NI-DMM

Setting Properties Before Reading Properties

Note: Not supported on the PCMCIA-4050 or the PXI/PCI-4060.

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:** .nidmmconfig

initiate

nidmm.Session.**initiate**()

Initiates an acquisition. After you call this method, the DMM leaves the Idle state and enters the Wait-for-Trigger state. If trigger is set to Immediate mode, the DMM begins acquiring measurement data. Use *nidmm.Session.fetch()*, *nidmm.Session.fetch_multi_point()*, or *nidmm.Session.fetch_waveform()* to retrieve the measurement data.

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

---

lock

nidmm.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 *nidmm.Session.lock()* method.
- A call to NI-DMM locked the session.
- After a call to the *nidmm.Session.lock()* method returns successfully, no other threads can access the device session until you call the *nidmm.Session.unlock()* method or exit out of the with block when using lock context manager.
- Use the *nidmm.Session.lock()* method and the *nidmm.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 *nidmm.Session.lock()* method within the same thread. To completely unlock the session, you must balance each call to the *nidmm.Session.lock()* method with a call to the *nidmm.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 nidmm.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, *nidmm.Session.lock()* acts as a context manager and unlock will be called when the *with* block is exited

perform_open_cable_comp

nidmm.Session.**perform_open_cable_comp**()

For the NI 4082 and NI 4072 only, performs the open cable compensation measurements for the current capacitance/inductance range, and returns open cable compensation **Conductance** and **Susceptance** values. You can use the return values of this method as inputs to `nidmm.Session.ConfigureOpenCableCompValues()`.

This method returns an error if the value of the `nidmm.Session.method` property is not set to `CAPACITANCE` (1005) or `INDUCTANCE` (1006).

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

Return type

tuple (conductance, susceptance)

WHERE

conductance (float):

conductance is the measured value of open cable compensation **conductance**.

susceptance (float):

susceptance is the measured value of open cable compensation **susceptance**.

perform_short_cable_comp

nidmm.Session.**perform_short_cable_comp**()

Performs the short cable compensation measurements for the current capacitance/inductance range, and returns short cable compensation **Resistance** and **Reactance** values. You can use the return values of this method as inputs to `nidmm.Session.ConfigureShortCableCompValues()`.

This method returns an error if the value of the `nidmm.Session.method` property is not set to `CAPACITANCE` (1005) or `INDUCTANCE` (1006).

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

Return type

tuple (resistance, reactance)

WHERE

resistance (float):

resistance is the measured value of short cable compensation **resistance**.

reactance (float):

reactance is the measured value of short cable compensation **reactance**.

read

nidmm.Session.**read**(*maximum_time=hightime.timedelta(milliseconds=-1)*)

Acquires a single measurement and returns the measured value.

Parameters

maximum_time (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the **maximum_time** allowed for this method to complete in milliseconds. If the method does not complete within this time interval, the method returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.

The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.

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.

Return type

float

Returns

The measured value returned from the DMM.

read_multi_point

nidmm.Session.**read_multi_point**(*array_size*, *maximum_time=hightime.timedelta(milliseconds=-1)*)

Acquires multiple measurements and returns an array of measured values. The number of measurements the DMM makes is determined by the values you specify for the **Trigger_Count** and **Sample_Count** parameters in *nidmm.Session.configure_multi_point()*.

Parameters

- **array_size** (*int*) – Specifies the number of measurements to acquire. The maximum number of measurements for a finite acquisition is the (**Trigger Count** x **Sample Count**) parameters in *nidmm.Session.configure_multi_point()*.

  For continuous acquisitions, up to 100,000 points can be returned at once. The number of measurements can be a subset. The valid range is any positive ViInt32. The default value is 1.

- **maximum_time** (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the **maximum_time** allowed for this method to complete in milliseconds. If the method does not complete within this time interval, the method returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.

  The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.

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.

Return type
array.array("d")

Returns

An array of measurement values.

Note: The size of the **Reading_Array** must be at least the size that you specify for the **Array_Size** parameter.

read_status

nidmm.Session.**read_status**()

Returns measurement backlog and acquisition status. Use this method to determine how many measurements are available before calling *nidmm.Session.fetch()*, *nidmm.Session.fetch_multi_point()*, or *nidmm.Session.fetch_waveform()*.

Note: The NI 4050 is not supported.

Return type

tuple (acquisition_backlog, acquisition_status)

WHERE

acquisition_backlog (int):

The number of measurements available to be read. If the backlog continues to increase, data is eventually overwritten, resulting in an error.

Note: On the NI 4060, the **Backlog** does not increase when autoranging. On the NI 4065, the **Backlog** does not increase when Range is set to AUTO RANGE ON (-1), or before the first point is fetched when Range is set to AUTO RANGE ONCE (-3). These behaviors are due to the autorange model of the devices.

acquisition_status (*nidmm.AcquisitionStatus*):

Indicates status of the acquisition. The following table shows the acquisition states:

0	Running
1	Finished with backlog
2	Finished with no backlog
3	Paused
4	No acquisition in progress

read_waveform

nidmm.Session.**read_waveform**(*array_size*, *maximum_time=hightime.timedelta(milliseconds=-1)*)

For the NI 4080/4081/4082 and the NI 4070/4071/4072, acquires a waveform and returns data as an array of values or as a waveform data type. The number of elements in the **Waveform_Array** is determined by the values you specify for the **Waveform_Points** parameter in *nidmm.Session.configure_waveform_acquisition()*.

Parameters

- **array_size** (*int*) – Specifies the number of waveform points to return. You specify the total number of points that the DMM acquires in the **Waveform Points** parameter of *nidmm.Session.configure_waveform_acquisition()*. The default value is 1.
- **maximum_time** (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the **maximum_time** allowed for this method to complete in milliseconds. If the method does not complete within this time interval, the method returns the NIDMM_ERROR_MAX_TIME_EXCEEDED error code. This may happen if an external trigger has not been received, or if the specified timeout is not long enough for the acquisition to complete.

  The valid range is 0–86400000. The default value is NIDMM_VAL_TIME_LIMIT_AUTO (-1). The DMM calculates the timeout automatically.

  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.

Return type
array.array("d")

Returns

An array of measurement values.

Note: The size of the **Waveform_Array** must be at least the size that you specify for the **Array_Size** parameter.

reset

nidmm.Session.**reset**()

Resets the instrument to a known state and sends initialization commands to the instrument. The initialization commands set instrument settings to the state necessary for the operation of the instrument driver.

reset_with_defaults

nidmm.Session.**reset_with_defaults**()

Resets the instrument to a known state and sends initialization commands to the DMM. The initialization commands set the DMM settings to the state necessary for the operation of NI-DMM. All user-defined default values associated with a logical name are applied after setting the DMM.

self_cal

nidmm.Session.**self_cal**()

For the NI 4080/4081/4082 and the NI 4070/4071/4072, executes the self-calibration routine to maintain measurement accuracy.

Note: This method calls *nidmm.Session.reset()*, and any configurations previous to the call will be lost. All properties will be set to their default values after the call returns.

---

self_test

nidmm.Session.**self_test**()

Performs a self-test on the DMM to ensure that the DMM is functioning properly. Self-test does not calibrate the DMM. Zero indicates success.

On the NI 4080/4082 and NI 4070/4072, the error code 1013 indicates that you should check the fuse and replace it, if necessary.

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

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

Note: Self-test does not check the fuse on the NI 4065, NI 4071, and NI 4081. Hence, even if the fuse is blown on the device, self-test does not return error code 1013.

Note: This method calls *nidmm.Session.reset()*, and any configurations previous to the call will be lost. All properties will be set to their default values after the call returns.

---

send_software_trigger

nidmm.Session.**send_software_trigger**()

Sends a command to trigger the DMM. Call this method if you have configured either the *nidmm.Session.trigger_source* or *nidmm.Session.sample_trigger* properties. If the *nidmm.Session.trigger_source* and/or *nidmm.Session.sample_trigger* properties are set to NIDMM_VAL_EXTERNAL or NIDMM_VAL_TTL*n*, you can use this method to override the trigger source that you configured and trigger the device. The NI 4050 and NI 4060 are not supported.

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.

unlock

nidmm.Session.**unlock**()

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

Properties

ac_max_freq

nidmm.Session.**ac_max_freq**

Specifies the maximum frequency component of the input signal for AC measurements. This property is used only for error checking and verifies that the value of this parameter is less than the maximum frequency of the device. This property affects the DMM only when you set the `nidmm.Session.method` property to AC measurements. The valid range is 1 Hz-300 kHz for the NI 4070/4071/4072, 10 Hz-100 kHz for the NI 4065, and 20 Hz-25 kHz for the NI 4050 and NI 4060.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Max Frequency**
- C Attribute: **NIDMM_ATTR_AC_MAX_FREQ**

ac_min_freq

nidmm.Session.**ac_min_freq**

Specifies the minimum frequency component of the input signal for AC measurements. This property affects the DMM only when you set the `nidmm.Session.method` property to AC measurements. The valid range is 1 Hz-300 kHz for the NI 4070/4071/4072, 10 Hz-100 kHz for the NI 4065, and 20 Hz-25 kHz for the NI 4050 and NI 4060.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Min Frequency**
- C Attribute: **NIDMM_ATTR_AC_MIN_FREQ**

adc_calibration

nidmm.Session.**adc_calibration**

For the NI 4070/4071/4072 only, specifies the ADC calibration mode.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ADCCalibration
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:ADC Calibration**
- C Attribute: **NIDMM_ATTR_ADC_CALIBRATION**

aperture_time

nidmm.Session.**aperture_time**

Specifies the measurement aperture time for the current configuration. Aperture time is specified in units set by *nidmm.Session.aperture_time_units*. To override the default aperture, set this property to the desired aperture time after calling `nidmm.Session.ConfigureMeasurement()`. To return to the default, set this property to `NIDMM_VAL_APERTURE_TIME_AUTO` (-1). On the NI 4070/4071/4072, the minimum aperture time is 8.89 usec, and the maximum aperture time is 149 sec. Any number of powerline cycles (PLCs) within the minimum and maximum ranges is allowed on the NI 4070/4071/4072. On the NI 4065 the minimum aperture time is 333 µs, and the maximum aperture time is 78.2 s. If setting the number of averages directly, the total measurement time is aperture time X the number of averages, which must be less than 72.8 s. The aperture times allowed are 333 µs, 667 µs, or multiples of 1.11 ms-for example 1.11 ms, 2.22 ms, 3.33 ms, and so on. If you set an aperture time other than 333 µs, 667 µs, or multiples of 1.11 ms, the value will be coerced up to the next supported aperture time. On the NI 4060, when the powerline frequency is 60 Hz, the PLCs allowed are 1 PLC, 6 PLC, 12 PLC, and 120 PLC. When the powerline frequency is 50 Hz, the PLCs allowed are 1 PLC, 5 PLC, 10 PLC, and 100 PLC.

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

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.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Advanced:Aperture Time**
- C Attribute: **NIDMM_ATTR_APERTURE_TIME**

aperture_time_units

nidmm.Session.**aperture_time_units**

Specifies the units of aperture time for the current configuration. The NI 4060 does not support an aperture time set in seconds.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ApertureTimeUnits
Permissions	read-write

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

- LabVIEW Property: **Configuration:Advanced:Aperture Time Units**
- C Attribute: **NIDMM_ATTR_APERTURE_TIME_UNITS**

auto_range_value

nidmm.Session.**auto_range_value**

Specifies the value of the range. If auto ranging, shows the actual value of the active range. The value of this property is set during a read operation.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read only

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

- LabVIEW Property: **Configuration:Auto Range Value**

- C Attribute: **NIDMM_ATTR_AUTO_RANGE_VALUE**

auto_zero

nidmm.Session.**auto_zero**

Specifies the AutoZero mode. The NI 4050 is not supported.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.AutoZero
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Auto Zero**
- C Attribute: **NIDMM_ATTR_AUTO_ZERO**

buffer_size

nidmm.Session.**buffer_size**

Size in samples of the internal data buffer. Maximum is 134,217,727 (OX7FFFFFF) samples. When set to NIDMM_VAL_BUFFER_SIZE_AUTO (-1), NI-DMM chooses the buffer size.

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.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Multi Point Acquisition:Advanced:Buffer Size**
- C Attribute: **NIDMM_ATTR_BUFFER_SIZE**

cable_comp_type

nidmm.Session.**cable_comp_type**

For the NI 4072 only, the type of cable compensation that is applied to the current capacitance or inductance measurement for the current range. Changing the method or the range through this property or through *nidmm.Session.configure_measurement_digits()* resets the value of this property to the default value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.CableCompensationType
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Cable Compensation Type**
- C Attribute: **NIDMM_ATTR_CABLE_COMP_TYPE**

channel_count

nidmm.Session.**channel_count**

Indicates the number of channels that the specific instrument driver supports. For each property for which the IVI_VAL_MULTI_CHANNEL flag property is set, 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

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

- LabVIEW Property: **Inherent IVI Attributes:Instrument Capabilities:Channel Count**
- C Attribute: **NIDMM_ATTR_CHANNEL_COUNT**

current_source

nidmm.Session.**current_source**

Specifies the current source provided during diode measurements. The NI 4050 and NI 4060 are not supported.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Current Source**
- C Attribute: **NIDMM_ATTR_CURRENT_SOURCE**

dc_bias

nidmm.Session.**dc_bias**

For the NI 4072 only, controls the available DC bias for capacitance measurements.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Advanced:DC Bias**
- C Attribute: **NIDMM_ATTR_DC_BIAS**

dc_noise_rejection

nidmm.Session.**dc_noise_rejection**

Specifies the DC noise rejection mode. The NI 4050 and NI 4060 are not supported.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.DCNoiseRejection
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:DC Noise Rejection**
- C Attribute: **NIDMM_ATTR_DC_NOISE_REJECTION**

driver_setup

nidmm.Session.**driver_setup**

This property indicates the Driver Setup string that the user specified when initializing the driver. Some cases exist where the end-user must specify instrument driver options at initialization time. An example of this is specifying a particular instrument model from among a family of instruments that the driver supports. This is useful when using simulation. The end-user can specify driver-specific options through the DriverSetup keyword in the optionsString parameter to the niDMM Init With Options.vi. If the user does 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

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

- LabVIEW Property: **Inherent IVI Attributes:User Options:Driver Setup**
- C Attribute: **NIDMM_ATTR_DRIVER_SETUP**

freq_voltage_auto_range

nidmm.Session.**freq_voltage_auto_range**

For the NI 4070/4071/4072 only, specifies the value of the frequency voltage range. If Auto Ranging, shows the actual value of the active frequency voltage range. If not Auto Ranging, the value of this property is the same as that of *nidmm.Session.freq_voltage_range*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read only

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

- LabVIEW Property: **Configuration:Measurement Options:Frequency Voltage Auto Range Value**
- C Attribute: **NIDMM_ATTR_FREQ_VOLTAGE_AUTO_RANGE**

freq_voltage_range

nidmm.Session.**freq_voltage_range**

Specifies the maximum amplitude of the input signal for frequency measurements.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Frequency Voltage Range**
- C Attribute: **NIDMM_ATTR_FREQ_VOLTAGE_RANGE**

function

nidmm.Session.**function**

Specifies the measurement method. Refer to the `nidmm.Session.method` topic in the NI Digital Multimeters Help for device-specific information. If you are setting this property directly, you must also set the *nidmm.Session.operation_mode* property, which controls whether the DMM takes standard single or multipoint measurements, or acquires a waveform. If you are programming properties directly, you must set the *nidmm.Session.operation_mode* property before setting other configuration properties. If the *nidmm.Session.operation_mode* property is set to *WAVEFORM*, the only valid method types are `WAVEFORM_VOLTAGE` and `WAVEFORM_CURRENT`. Set the *nidmm.Session.operation_mode* property to *IVIDMM* to set all other method values.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.Function
Permissions	read-write

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

- LabVIEW Property: **Configuration:Function**
- C Attribute: **NIDMM_ATTR_FUNCTION**

input_resistance

nidmm.Session.**input_resistance**

Specifies the input resistance of the instrument. The NI 4050 and NI 4060 are not supported.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Input Resistance**
- C Attribute: **NIDMM_ATTR_INPUT_RESISTANCE**

instrument_firmware_revision

nidmm.Session.**instrument_firmware_revision**

A string containing the instrument firmware revision number.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Instrument Firmware Revision**
- C Attribute: **NIDMM_ATTR_INSTRUMENT_FIRMWARE_REVISION**

instrument_manufacturer

nidmm.Session.**instrument_manufacturer**

A string containing the manufacturer of the instrument.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Instrument Manufacturer**
- C Attribute: **NIDMM_ATTR_INSTRUMENT_MANUFACTURER**

instrument_model

nidmm.Session.**instrument_model**

A string containing the instrument model.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Instrument Model**
- C Attribute: **NIDMM_ATTR_INSTRUMENT_MODEL**

instrument_product_id

nidmm.Session.**instrument_product_id**

The PCI product ID.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Instrument Product ID**
- C Attribute: **NIDMM_ATTR_INSTRUMENT_PRODUCT_ID**

io_resource_descriptor

nidmm.Session.**io_resource_descriptor**

A string containing the resource descriptor of the instrument.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Advanced Session Information:I/O Resource Descriptor**
- C Attribute: **NIDMM_ATTR_IO_RESOURCE_DESCRIPTOR**

lc_calculation_model

nidmm.Session.**lc_calculation_model**

For the NI 4072 only, specifies the type of algorithm that the measurement processing uses for capacitance and inductance measurements.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.LCCalculationModel
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Advanced:Calculation Model**
- C Attribute: **NIDMM_ATTR_LC_CALCULATION_MODEL**

lc_number_meas_to_average

nidmm.Session.**lc_number_meas_to_average**

For the NI 4072 only, specifies the number of LC measurements that are averaged to produce one reading.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Number of LC Measurements To Average**
- C Attribute: **NIDMM_ATTR_LC_NUMBER_MEAS_TO_AVERAGE**

logical_name

`nidmm.Session.`**`logical_name`**

A string containing the logical name of the instrument.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Advanced Session Information:Logical Name**
- C Attribute: **NIDMM_ATTR_LOGICAL_NAME**

meas_complete_dest

`nidmm.Session.`**`meas_complete_dest`**

Specifies the destination of the measurement complete (MC) signal. The NI 4050 is not supported. To determine which values are supported by each device, refer to the LabWindows/CVI Trigger Routing section in the NI Digital Multimeters Help.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.MeasurementCompleteDest
Permissions	read-write

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

- LabVIEW Property: **Trigger:Measurement Complete Dest**
- C Attribute: **NIDMM_ATTR_MEAS_COMPLETE_DEST**

number_of_averages

nidmm.Session.**number_of_averages**

Specifies the number of averages to perform in a measurement. For the NI 4070/4071/4072, applies only when the aperture time is not set to AUTO and Auto Zero is ON. The default is 1. The NI 4050 and NI 4060 are not supported.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Configuration:Advanced:Number Of Averages**
- C Attribute: **NIDMM_ATTR_NUMBER_OF_AVERAGES**

offset_comp_ohms

nidmm.Session.**offset_comp_ohms**

For the NI 4070/4071/4072 only, enables or disables offset compensated ohms.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Offset Compensated Ohms**
- C Attribute: **NIDMM_ATTR_OFFSET_COMP_OHMS**

open_cable_comp_conductance

nidmm.Session.**open_cable_comp_conductance**

For the NI 4072 only, specifies the active part (conductance) of the open cable compensation. The valid range is any real number greater than 0. The default value (-1.0) indicates that compensation has not taken place. Changing the method or the range through this property or through *nidmm.Session.configure_measurement_digits()* resets the value of this property to the default value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Open Cable Compensation Values:Conductance**
- C Attribute: **NIDMM_ATTR_OPEN_CABLE_COMP_CONDUCTANCE**

open_cable_comp_susceptance

nidmm.Session.**open_cable_comp_susceptance**

For the NI 4072 only, specifies the reactive part (susceptance) of the open cable compensation. The valid range is any real number greater than 0. The default value (-1.0) indicates that compensation has not taken place. Changing the method or the range through this property or through *nidmm.Session.configure_measurement_digits()* resets the value of this property to the default value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Open Cable Compensation Values:Susceptance**
- C Attribute: **NIDMM_ATTR_OPEN_CABLE_COMP_SUSCEPTANCE**

operation_mode

nidmm.Session.**operation_mode**

Specifies how the NI 4065 and NI 4070/4071/4072 acquire data. When you call *nidmm.Session.configure_measurement_digits()*, NI-DMM sets this property to *IVIDMM*. When you call *nidmm.Session.configure_waveform_acquisition()*, NI-DMM sets this property to *WAVEFORM*. If you are programming properties directly, you must set this property before setting other configuration properties.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.OperationMode
Permissions	read-write

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

- LabVIEW Property: **Configuration:Advanced:Operation Mode**
- C Attribute: **NIDMM_ATTR_OPERATION_MODE**

powerline_freq

nidmm.Session.**powerline_freq**

Specifies the powerline frequency. The NI 4050 and NI 4060 use this value to select an aperture time to reject powerline noise by selecting the appropriate internal sample clock and filter. The NI 4065 and NI 4070/4071/4072 use this value to select a timebase for setting the *nidmm.Session.aperture_time* property in powerline cycles (PLCs). After configuring powerline frequency, set the *nidmm.Session.aperture_time_units* property to PLCs. When setting the *nidmm.Session.aperture_time* property, select the number of PLCs for the powerline frequency. For example, if powerline frequency = 50 Hz (or 20ms) and aperture time in PLCs = 5, then aperture time in Seconds = 20ms * 5 PLCs = 100 ms. Similarly, if powerline frequency = 60 Hz (or 16.667 ms) and aperture time in PLCs = 6, then aperture time in Seconds = 16.667 ms * 6 PLCs = 100 ms.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Powerline Frequency**
- C Attribute: **NIDMM_ATTR_POWERLINE_FREQ**

range

nidmm.Session.**range**

Specifies the measurement range. Use positive values to represent the absolute value of the maximum expected measurement. The value is in units appropriate for the current value of the `nidmm.Session.method` property. For example, if `nidmm.Session.method` is set to `NIDMM_VAL_VOLTS`, the units are volts. The NI 4050 and NI 4060 only support Auto Range when the trigger and sample trigger is set to IMMEDIATE. `NIDMM_VAL_AUTO_RANGE_ON` -1.0 NI-DMM performs an Auto Range before acquiring the measurement. `NIDMM_VAL_AUTO_RANGE_OFF` -2.0 NI-DMM sets the Range to the current *nidmm.Session.auto_range_value* and uses this range for all subsequent measurements until the measurement configuration is changed. `NIDMM_VAL_AUTO_RANGE_ONCE` -3.0 NI-DMM performs an Auto Range before acquiring the next measurement. The *nidmm.Session.auto_range_value* is stored and used for all subsequent measurements until the measurement configuration is changed.

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.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Range**
- C Attribute: **NIDMM_ATTR_RANGE**

---

resolution_absolute

nidmm.Session.**resolution_absolute**

Specifies the measurement resolution in absolute units. Setting this property to higher values increases the measurement accuracy. Setting this property to lower values increases the measurement speed. NI-DMM ignores this property for capacitance and inductance measurements on the NI 4072. To achieve better resolution for such measurements, use the *nidmm.Session.lc_number_meas_to_average* property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Absolute Resolution**
- C Attribute: **NIDMM_ATTR_RESOLUTION_ABSOLUTE**

resolution_digits

nidmm.Session.**resolution_digits**

Specifies the measurement resolution in digits. Setting this property to higher values increases the measurement accuracy. Setting this property to lower values increases the measurement speed. NI-DMM ignores this property for capacitance and inductance measurements on the NI 4072. To achieve better resolution for such measurements, use the *nidmm.Session.lc_number_meas_to_average* property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Digits Resolution**
- C Attribute: **NIDMM_ATTR_RESOLUTION_DIGITS**

sample_count

nidmm.Session.**sample_count**

Specifies the number of measurements the DMM takes each time it receives a trigger in a multiple point acquisition.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Multi Point Acquisition:Sample Count**
- C Attribute: **NIDMM_ATTR_SAMPLE_COUNT**

sample_interval

nidmm.Session.**sample_interval**

Specifies the amount of time in seconds the DMM waits between measurement cycles. This property only applies when the *nidmm.Session.sample_trigger* property is set to INTERVAL. On the NI 4060, the value for this property is used as the settling time. When this property is set to 0, the NI 4060 does not settle between measurement cycles. The onboard timing resolution is 1 µs on the NI 4060. The NI 4065 and NI 4070/4071/4072 use the value specified in this property as additional delay. On the NI 4065 and NI 4070/4071/4072, the onboard timing resolution is 34.72 ns and the valid range is 0-149 s. Only positive values are valid when setting the sample interval. The NI 4050 is not supported.

The following table lists the characteristics of this property.

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

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

- LabVIEW Property: **Multi Point Acquisition:Sample Interval**
- C Attribute: **NIDMM_ATTR_SAMPLE_INTERVAL**

sample_trigger

nidmm.Session.**sample_trigger**

Specifies the sample trigger source. To determine which values are supported by each device, refer to the LabWindows/CVI Trigger Routing section in the NI Digital Multimeters Help.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.SampleTrigger
Permissions	read-write

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

- LabVIEW Property: **Multi Point Acquisition:Sample Trigger**
- C Attribute: **NIDMM_ATTR_SAMPLE_TRIGGER**

serial_number

nidmm.Session.**serial_number**

A string containing the serial number of the instrument. This property corresponds to the serial number label that is attached to most products.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Instrument Identification:Instrument Serial Number**
- C Attribute: **NIDMM_ATTR_SERIAL_NUMBER**

settle_time

nidmm.Session.**settle_time**

Specifies the settling time in seconds. To override the default settling time, set this property. To return to the default, set this property to `NIDMM_VAL_SETTLE_TIME_AUTO` (-1). The NI 4050 and NI 4060 are not supported.

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.

The following table lists the characteristics of this property.

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

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

- LabVIEW Property: **Configuration:Advanced:Settle Time**
- C Attribute: **NIDMM_ATTR_SETTLE_TIME**

short_cable_comp_reactance

nidmm.Session.**short_cable_comp_reactance**

For the NI 4072 only, represents the reactive part (reactance) of the short cable compensation. The valid range is any real number greater than 0. The default value (-1) indicates that compensation has not taken place. Changing the method or the range through this property or through *nidmm.Session.configure_measurement_digits()* resets the value of this property to the default value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Short Cable Compensation Values:Reactance**
- C Attribute: **NIDMM_ATTR_SHORT_CABLE_COMP_REACTANCE**

short_cable_comp_resistance

nidmm.Session.**short_cable_comp_resistance**

For the NI 4072 only, represents the active part (resistance) of the short cable compensation. The valid range is any real number greater than 0. The default value (-1) indicates that compensation has not taken place. Changing the method or the range through this property or through *nidmm.Session.configure_measurement_digits()* resets the value of this property to the default value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Capacitance and Inductance:Short Cable Compensation Values:Resistance**
- C Attribute: **NIDMM_ATTR_SHORT_CABLE_COMP_RESISTANCE**

simulate

nidmm.Session.**simulate**

Specifies whether or not to simulate instrument driver I/O operations. If simulation is enabled, instrument driver methods perform range checking and call IVI Get and Set methods, but they do not perform instrument I/O. For output parameters that represent instrument data, the instrument driver methods return calculated values. The default value is False (0). Use the `nidmm.Session.__init__()` method to override this setting. Simulate can only be set within the InitWithOptions method. The property value cannot be changed outside of the method.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write

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

- LabVIEW Property: **Inherent IVI Attributes:User Options:Simulate**
- C Attribute: **NIDMM_ATTR_SIMULATE**

specific_driver_description

nidmm.Session.**specific_driver_description**

A string containing a description of the specific driver.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Specific Driver Identification:Specific Driver Description**
- C Attribute: **NIDMM_ATTR_SPECIFIC_DRIVER_DESCRIPTION**

specific_driver_major_version

nidmm.Session.**specific_driver_major_version**

Returns the major version number of this instrument driver.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Version Info:Specific Driver Major Version**
- C Attribute: **NIDMM_ATTR_SPECIFIC_DRIVER_MAJOR_VERSION**

specific_driver_minor_version

nidmm.Session.**specific_driver_minor_version**

The minor version number of this instrument driver.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Version Info:Specific Driver Minor Version**
- C Attribute: **NIDMM_ATTR_SPECIFIC_DRIVER_MINOR_VERSION**

specific_driver_revision

nidmm.Session.**specific_driver_revision**

A string that contains additional version information about this specific instrument driver.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Version Info:Specific Driver Revision**
- C Attribute: **NIDMM_ATTR_SPECIFIC_DRIVER_REVISION**

specific_driver_vendor

nidmm.Session.**specific_driver_vendor**

A string containing the vendor of the specific driver.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Specific Driver Identification:Specific Driver Vendor**
- C Attribute: **NIDMM_ATTR_SPECIFIC_DRIVER_VENDOR**

supported_instrument_models

nidmm.Session.**supported_instrument_models**

A string containing the instrument models supported by the specific driver.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only

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

- LabVIEW Property: **Inherent IVI Attributes:Specific Driver Capabilities:Supported Instrument Models**
- C Attribute: **NIDMM_ATTR_SUPPORTED_INSTRUMENT_MODELS**

temp_rtd_a

nidmm.Session.**temp_rtd_a**

Specifies the Callendar-Van Dusen A coefficient for RTD scaling when the RTD Type property is set to Custom. The default value is 3.9083e-3 (Pt3851).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Resistance Temperature Detector:RTD A**
- C Attribute: **NIDMM_ATTR_TEMP_RTD_A**

temp_rtd_b

nidmm.Session.**temp_rtd_b**

Specifies the Callendar-Van Dusen B coefficient for RTD scaling when the RTD Type property is set to Custom. The default value is -5.775e-7(Pt3851).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Resistance Temperature Detector:RTD B**
- C Attribute: **NIDMM_ATTR_TEMP_RTD_B**

temp_rtd_c

nidmm.Session.**temp_rtd_c**

Specifies the Callendar-Van Dusen C coefficient for RTD scaling when the RTD Type property is set to Custom. The default value is -4.183e-12(Pt3851).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Resistance Temperature Detector:RTD C**
- C Attribute: **NIDMM_ATTR_TEMP_RTD_C**

temp_rtd_res

nidmm.Session.**temp_rtd_res**

Specifies the RTD resistance at 0 degrees Celsius. This applies to all supported RTDs, including custom RTDs. The default value is 100 (?).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Resistance Temperature Detector:RTD Resistance**
- C Attribute: **NIDMM_ATTR_TEMP_RTD_RES**

temp_rtd_type

nidmm.Session.**temp_rtd_type**

Specifies the type of RTD used to measure temperature. The default value is *PT3851*. Refer to the *nidmm.Session.temp_rtd_type* topic in the NI Digital Multimeters Help for additional information about defined values.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.RTDType
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Resistance Temperature Detector:RTD Type**
- C Attribute: **NIDMM_ATTR_TEMP_RTD_TYPE**

temp_tc_fixed_ref_junc

nidmm.Session.**temp_tc_fixed_ref_junc**

Specifies the reference junction temperature when a fixed reference junction is used to take a thermocouple measurement. The default value is 25.0 (°C).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Thermocouple:Fixed Reference Junction**
- C Attribute: **NIDMM_ATTR_TEMP_TC_FIXED_REF_JUNC**

temp_tc_ref_junc_type

nidmm.Session.**temp_tc_ref_junc_type**

Specifies the type of reference junction to be used in the reference junction compensation of a thermocouple. The only supported value, *FIXED*, is fixed.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ThermocoupleReferenceJunctionType
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Thermocouple:Reference Junction Type**
- C Attribute: **NIDMM_ATTR_TEMP_TC_REF_JUNC_TYPE**

temp_tc_type

nidmm.Session.**temp_tc_type**

Specifies the type of thermocouple used to measure the temperature. The default value is *J*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ThermocoupleType
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Thermocouple:Thermocouple Type**
- C Attribute: **NIDMM_ATTR_TEMP_TC_TYPE**

temp_thermistor_a

nidmm.Session.**temp_thermistor_a**

Specifies the Steinhart-Hart A coefficient for thermistor scaling when the Thermistor Type property is set to Custom. The default value is 0.0010295 (44006).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Thermistor:Thermistor A**
- C Attribute: **NIDMM_ATTR_TEMP_THERMISTOR_A**

temp_thermistor_b

nidmm.Session.**temp_thermistor_b**

Specifies the Steinhart-Hart B coefficient for thermistor scaling when the Thermistor Type proerty is set to Custom. The default value is 0.0002391 (44006).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Thermistor:Thermistor B**
- C Attribute: **NIDMM_ATTR_TEMP_THERMISTOR_B**

temp_thermistor_c

nidmm.Session.**temp_thermistor_c**

Specifies the Steinhart-Hart C coefficient for thermistor scaling when the Thermistor Type property is set to Custom. The default value is 1.568e-7 (44006).

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Thermistor:Thermistor C**
- C Attribute: **NIDMM_ATTR_TEMP_THERMISTOR_C**

temp_thermistor_type

nidmm.Session.**temp_thermistor_type**

Specifies the type of thermistor used to measure the temperature. The default value is *THERMISTOR_44006*. Refer to the *nidmm.Session.temp_thermistor_type* topic in the NI Digital Multimeters Help for additional information about defined values.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ThermistorType
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Thermistor:Thermistor Type**
- C Attribute: **NIDMM_ATTR_TEMP_THERMISTOR_TYPE**

temp_transducer_type

nidmm.Session.**temp_transducer_type**

Specifies the type of device used to measure the temperature. The default value is NIDMM_VAL_4_THERMOCOUPLE.

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.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TransducerType
Permissions	read-write

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

- LabVIEW Property: **Configuration:Measurement Options:Temperature:Transducer Type**
- C Attribute: **NIDMM_ATTR_TEMP_TRANSDUCER_TYPE**

trigger_count

nidmm.Session.**trigger_count**

Specifies the number of triggers the DMM receives before returning to the Idle state. This property can be set to any positive ViInt32 value for the NI 4065 and NI 4070/4071/4072. The NI 4050 and NI 4060 support this property being set to 1. Refer to the Multiple Point Acquisitions section of the NI Digital Multimeters Help for more information.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Multi Point Acquisition:Trigger Count**
- C Attribute: **NIDMM_ATTR_TRIGGER_COUNT**

trigger_delay

nidmm.Session.**trigger_delay**

Specifies the time (in seconds) that the DMM waits after it has received a trigger before taking a measurement. The default value is AUTO DELAY (-1), which means that the DMM waits an appropriate settling time before taking the measurement. (-1) signifies that AUTO DELAY is on, and (-2) signifies that AUTO DELAY is off. The NI 4065 and NI 4070/4071/4072 use the value specified in this property as additional settling time. For the The NI 4065 and NI 4070/4071/4072, the valid range for Trigger Delay is AUTO DELAY (-1) or 0.0-149.0 seconds and the onboard timing resolution is 34.72 ns. On the NI 4060, if this property is set to 0, the DMM does not settle before taking the measurement. On the NI 4060, the valid range for AUTO DELAY (-1) is 0.0-12.0 seconds and the onboard timing resolution is 100 ms. When using the NI 4050, this property must be set to AUTO DELAY (-1). Use positive values to set the trigger delay in seconds. Valid Range: `NIDMM_VAL_AUTO_DELAY` (-1.0), 0.0-12.0 seconds (NI 4060 only) Default Value: `NIDMM_VAL_AUTO_DELAY`

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.

The following table lists the characteristics of this property.

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

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

- LabVIEW Property: **Trigger:Trigger Delay**
- C Attribute: **NIDMM_ATTR_TRIGGER_DELAY**

---

trigger_source

nidmm.Session.**trigger_source**

Specifies the trigger source. When `nidmm.Session._initiate()` is called, the DMM waits for the trigger specified with this property. After it receives the trigger, the DMM waits the length of time specified with the *nidmm.Session.trigger_delay* property. The DMM then takes a measurement. This property is not supported on the NI 4050. To determine which values are supported by each device, refer to the LabWindows/CVI Trigger Routing section in the NI Digital Multimeters Help.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.TriggerSource
Permissions	read-write

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

- LabVIEW Property: **Trigger:Trigger Source**
- C Attribute: **NIDMM_ATTR_TRIGGER_SOURCE**

waveform_coupling

nidmm.Session.**waveform_coupling**

For the NI 4070/4071/4072 only, specifies the coupling during a waveform acquisition.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.WaveformCoupling
Permissions	read-write

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

- LabVIEW Property: **Waveform Acquisition:Waveform Coupling**
- C Attribute: **NIDMM_ATTR_WAVEFORM_COUPLING**

waveform_points

nidmm.Session.**waveform_points**

For the NI 4070/4071/4072 only, specifies the number of points to acquire in a waveform acquisition.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write

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

- LabVIEW Property: **Waveform Acquisition:Waveform Points**
- C Attribute: **NIDMM_ATTR_WAVEFORM_POINTS**

waveform_rate

nidmm.Session.**waveform_rate**

For the NI 4070/4071/4072 only, specifies the rate of the waveform acquisition in Samples per second (S/s). The valid Range is 10.0-1,800,000 S/s. Values are coerced to the closest integer divisor of 1,800,000. The default value is 1,800,000.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write

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

- LabVIEW Property: **Waveform Acquisition:Waveform Rate**
- C Attribute: **NIDMM_ATTR_WAVEFORM_RATE**

Session

- *Session*
- *Methods*
  - *abort*
  - *close*
  - *configure_measurement_absolute*
  - *configure_measurement_digits*
  - *configure_multi_point*
  - *configure_rtd_custom*
  - *configure_rtd_type*
  - *configure_thermistor_custom*
  - *configure_thermocouple*
  - *configure_trigger*
  - *configure_waveform_acquisition*
  - *disable*
  - *export_attribute_configuration_buffer*
  - *export_attribute_configuration_file*
  - *fetch*
  - *fetch_multi_point*
  - *fetch_waveform*
  - *fetch_waveform_into*

- *get_cal_date_and_time*
- *get_dev_temp*
- *get_ext_cal_recommended_interval*
- *get_last_cal_temp*
- *get_self_cal_supported*
- *import_attribute_configuration_buffer*
- *import_attribute_configuration_file*
- *initiate*
- *lock*
- *perform_open_cable_comp*
- *perform_short_cable_comp*
- *read*
- *read_multi_point*
- *read_status*
- *read_waveform*
- *reset*
- *reset_with_defaults*
- *self_cal*
- *self_test*
- *send_software_trigger*
- *unlock*

• *Properties*

- *ac_max_freq*
- *ac_min_freq*
- *adc_calibration*
- *aperture_time*
- *aperture_time_units*
- *auto_range_value*
- *auto_zero*
- *buffer_size*
- *cable_comp_type*
- *channel_count*
- *current_source*
- *dc_bias*
- *dc_noise_rejection*

- *driver_setup*
- *freq_voltage_auto_range*
- *freq_voltage_range*
- *function*
- *input_resistance*
- *instrument_firmware_revision*
- *instrument_manufacturer*
- *instrument_model*
- *instrument_product_id*
- *io_resource_descriptor*
- *lc_calculation_model*
- *lc_number_meas_to_average*
- *logical_name*
- *meas_complete_dest*
- *number_of_averages*
- *offset_comp_ohms*
- *open_cable_comp_conductance*
- *open_cable_comp_susceptance*
- *operation_mode*
- *powerline_freq*
- *range*
- *resolution_absolute*
- *resolution_digits*
- *sample_count*
- *sample_interval*
- *sample_trigger*
- *serial_number*
- *settle_time*
- *short_cable_comp_reactance*
- *short_cable_comp_resistance*
- *simulate*
- *specific_driver_description*
- *specific_driver_major_version*
- *specific_driver_minor_version*
- *specific_driver_revision*

- *specific_driver_vendor*
- *supported_instrument_models*
- *temp_rtd_a*
- *temp_rtd_b*
- *temp_rtd_c*
- *temp_rtd_res*
- *temp_rtd_type*
- *temp_tc_fixed_ref_junc*
- *temp_tc_ref_junc_type*
- *temp_tc_type*
- *temp_thermistor_a*
- *temp_thermistor_b*
- *temp_thermistor_c*
- *temp_thermistor_type*
- *temp_transducer_type*
- *trigger_count*
- *trigger_delay*
- *trigger_source*
- *waveform_coupling*
- *waveform_points*
- *waveform_rate*

Enums

Enums used in NI-DMM

ADCCalibration

class nidmm.**ADCCalibration**

AUTO

The DMM enables or disables ADC calibration for you.

OFF

The DMM does not compensate for changes to the gain.

ON

The DMM measures an internal reference to calculate the correct gain for the measurement.

AcquisitionStatus

class nidmm.**AcquisitionStatus**

RUNNING

Running

FINISHED_WITH_BACKLOG

Finished with **Backlog**

FINISHED_WITH_NO_BACKLOG

Finished with no **Backlog**

PAUSED

Paused

NO_ACQUISITION_IN_PROGRESS

No acquisition in progress

ApertureTimeUnits

class nidmm.**ApertureTimeUnits**

SECONDS

Seconds

POWER_LINE_CYCLES

Powerline Cycles

AutoZero

class nidmm.**AutoZero**

AUTO

The drivers chooses the AutoZero setting based on the configured method and resolution.

OFF

Disables AutoZero.

ON

The DMM internally disconnects the input signal following each measurement and takes a zero reading. It then subtracts the zero reading from the preceding reading.

ONCE

The DMM internally disconnects the input signal for the first measurement and takes a zero reading. It then subtracts the zero reading from the first reading and the following readings.

CableCompensationType

class nidmm.**CableCompensationType**

NONE

No Cable Compensation

OPEN

Open Cable Compensation

SHORT

Short Cable Compensation

OPEN_AND_SHORT

Open and Short Cable Compensation

DCNoiseRejection

class nidmm.**DCNoiseRejection**

AUTO

The driver chooses the DC noise rejection setting based on the configured method and resolution.

NORMAL

NI-DMM weighs all samples equally.

SECOND_ORDER

NI-DMM weighs the samples taken in the middle of the aperture time more than samples taken at the beginning and the end of the measurement using a triangular weighing method.

HIGH_ORDER

NI-DMM weighs the samples taken in the middle of the aperture time more than samples taken at the beginning and the end of the measurement using a bell-curve weighing method.

Function

class nidmm.**Function**

DC_VOLTS

DC Voltage

AC_VOLTS

AC Voltage

DC_CURRENT

DC Current

AC_CURRENT

AC Current

TWO_WIRE_RES

2-Wire Resistance

FOUR_WIRE_RES
4-Wire Resistance

FREQ
Frequency

PERIOD
Period

TEMPERATURE
NI 4065, NI 4070/4071/4072, and NI 4080/4081/4182 supported.

AC_VOLTS_DC_COUPLED
AC Voltage with DC Coupling

DIODE
Diode

WAVEFORM_VOLTAGE
Waveform voltage

WAVEFORM_CURRENT
Waveform current

CAPACITANCE
Capacitance

INDUCTANCE
Inductance

LCCalculationModel

class nidmm.**LCCalculationModel**

AUTO
NI-DMM chooses the algorithm based on method and range

SERIES
NI-DMM uses the series impedance model to calculate capacitance and inductance

PARALLEL
NI-DMM uses the parallel admittance model to calculate capacitance and inductance

MeasurementCompleteDest

class nidmm.**MeasurementCompleteDest**

NONE
No Trigger

EXTERNAL
AUX I/O Connector

PXI_TRIG0
PXI Trigger Line 0

PXI_TRIG1
PXI Trigger Line 1

PXI_TRIG2
PXI Trigger Line 2

PXI_TRIG3
PXI Trigger Line 3

PXI_TRIG4
PXI Trigger Line 4

PXI_TRIG5
PXI Trigger Line 5

PXI_TRIG6
PXI Trigger Line 6

PXI_TRIG7
PXI Trigger Line 7

LBR_TRIG0
Internal Trigger Line of a PXI/SCXI Combination Chassis

OperationMode

class nidmm.**OperationMode**

IVIDMM
IviDmm Mode

WAVEFORM
Waveform acquisition mode

RTDType

class nidmm.**RTDType**

CUSTOM
Performs Callendar-Van Dusen RTD scaling with the user-specified A, B, and C coefficients.

PT3750
Performs scaling for a Pt 3750 RTD.

PT3851
Performs scaling for a Pt 3851 RTD.

PT3911
Performs scaling for a Pt 3911 RTD.

PT3916
Performs scaling for a Pt 3916 RTD.

PT3920
Performs scaling for a Pt 3920 RTD.

PT3928

Performs scaling for a Pt 3928 RTD.

SampleTrigger

class nidmm.**SampleTrigger**

IMMEDIATE

No Trigger

EXTERNAL

AUX I/O Connector Trigger Line 0

SOFTWARE_TRIG

Software Trigger

INTERVAL

Interval Trigger

PXI_TRIG0

PXI Trigger Line 0

PXI_TRIG1

PXI Trigger Line 1

PXI_TRIG2

PXI Trigger Line 2

PXI_TRIG3

PXI Trigger Line 3

PXI_TRIG4

PXI Trigger Line 4

PXI_TRIG5

PXI Trigger Line 5

PXI_TRIG6

PXI Trigger Line 6

PXI_TRIG7

PXI Trigger Line 7

PXI_STAR

PXI Star Trigger Line

AUX_TRIG1

AUX I/0 Connector Trigger Line 1

LBR_TRIG1

Internal Trigger Line of a PXI/SCXI Combination Chassis

ThermistorType

class nidmm.**ThermistorType**

CUSTOM
Custom

THERMISTOR_44004
44004

THERMISTOR_44006
44006

THERMISTOR_44007
44007

ThermocoupleReferenceJunctionType

class nidmm.**ThermocoupleReferenceJunctionType**

FIXED
Thermocouple reference juction is fixed at the user-specified temperature.

ThermocoupleType

class nidmm.**ThermocoupleType**

B
Thermocouple type B

E
Thermocouple type E

J
Thermocouple type J

K
Thermocouple type K

N
Thermocouple type N

R
Thermocouple type R

S
Thermocouple type S

T
Thermocouple type T

TransducerType

class nidmm.**TransducerType**

THERMOCOUPLE
Thermocouple

THERMISTOR
Thermistor

TWO_WIRE_RTD
2-wire RTD

FOUR_WIRE_RTD
4-wire RTD

TriggerSource

class nidmm.**TriggerSource**

IMMEDIATE
No Trigger

EXTERNAL
AUX I/O Connector Trigger Line 0

SOFTWARE_TRIG
Software Trigger

PXI_TRIG0
PXI Trigger Line 0

PXI_TRIG1
PXI Trigger Line 1

PXI_TRIG2
PXI Trigger Line 2

PXI_TRIG3
PXI Trigger Line 3

PXI_TRIG4
PXI Trigger Line 4

PXI_TRIG5
PXI Trigger Line 5

PXI_TRIG6
PXI Trigger Line 6

PXI_TRIG7
PXI Trigger Line 7

PXI_STAR
PXI Star Trigger Line

AUX_TRIG1

AUX I/O Connector Trigger Line 1

LBR_TRIG1

Internal Trigger Line of a PXI/SCXI Combination Chassis

WaveformCoupling

class nidmm.**WaveformCoupling**

AC

AC Coupled

DC

DC Coupled

Exceptions and Warnings

Error

exception nidmm.errors.**Error**

Base exception type that all NI-DMM exceptions derive from

DriverError

exception nidmm.errors.**DriverError**

An error originating from the NI-DMM driver

UnsupportedConfigurationError

exception nidmm.errors.**UnsupportedConfigurationError**

An error due to using this module in an usupported platform.

DriverNotInstalledError

exception nidmm.errors.**DriverNotInstalledError**

An error due to using this module without the driver runtime installed.

DriverTooOldError

exception nidmm.errors.**DriverTooOldError**

An error due to using this module with an older version of the NI-DMM driver runtime.

DriverTooNewError

exception nidmm.errors.**DriverTooNewError**

An error due to the NI-DMM driver runtime being too new for this module.

InvalidRepeatedCapabilityError

exception nidmm.errors.**InvalidRepeatedCapabilityError**

An error due to an invalid character in a repeated capability

SelfTestError

exception nidmm.errors.**SelfTestError**

An error due to a failed self-test

RpcError

exception nidmm.errors.**RpcError**

An error specific to sessions to the NI gRPC Device Server

DriverWarning

exception nidmm.errors.**DriverWarning**

A warning originating from the NI-DMM driver

Examples

You can download all nidmm examples for latest version here

nidmm_fetch_waveform.py

Listing 1: (nidmm_fetch_waveform.py)

```
#!/usr/bin/python

import argparse
import nidmm
import sys
import time


def example(resource_name, options, function, range, points, rate):
    with nidmm.Session(resource_name=resource_name, options=options) as session:
        session.configure_waveform_acquisition(measurement_function=nidmm.
↪Function[function], range=range, rate=rate, waveform_points=points)
        with session.initiate():
```

(continues on next page)

(continued from previous page)

```
            while True:
                time.sleep(0.1)
                backlog, acquisition_state = session.read_status()
                if acquisition_state == nidmm.AcquisitionStatus.FINISHED_WITH_NO_BACKLOG:
                    break
                measurements = session.fetch_waveform(array_size=backlog)
                print(measurements)


def _main(argsv):
    parser = argparse.ArgumentParser(description='Performs a waveform acquisition using the NI-DMM API.', formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2', help='Resource name of an NI digital multimeter.')
    parser.add_argument('-f', '--function', default='WAVEFORM_VOLTAGE', choices=nidmm.Function.__members__.keys(), type=str.upper, help='Measurement function.')
    parser.add_argument('-r', '--range', default=10, type=float, help='Measurement range.')
    parser.add_argument('-p', '--points', default=10, type=int, help='Specifies the number of points to acquire before the waveform acquisition completes.')
    parser.add_argument('-s', '--rate', default=1000, type=int, help='Specifies the rate of the acquisition in samples per second.')
    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.function, args.range, args.points, args.rate)


def main():
    _main(sys.argv[1:])


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '4082', 'BoardType': 'PXIe', }, }
    example('PXI1Slot2', options, 'WAVEFORM_VOLTAGE', 10, 10, 1000)


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:4082; BoardType:PXIe',]
    _main(cmd_line)


if __name__ == '__main__':
    main()


```

nidmm_measurement.py

Listing 2: (nidmm_measurement.py)

```
#!/usr/bin/python

import argparse
import nidmm
import sys


def example(resource_name, option_string, function, range, digits):
    with nidmm.Session(resource_name=resource_name, options=option_string) as session:
        session.configure_measurement_digits(measurement_function=nidmm.Function[function], range=range, resolution_digits=digits)
        print(session.read())


def _main(argsv):
    supported_functions = list(nidmm.Function.__members__.keys())
    parser = argparse.ArgumentParser(description='Performs a single measurement using the NI-DMM API.', formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2', help='Resource name of an NI digital multimeter.')
    parser.add_argument('-f', '--function', default=supported_functions[0], choices=supported_functions, type=str.upper, help='Measurement function.')
    parser.add_argument('-r', '--range', default=10, type=float, help='Measurement range.')
    parser.add_argument('-d', '--digits', default=6.5, type=float, help='Digits of resolution for the measurement.')
    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.function, args.range, args.digits)


def main():
    _main(sys.argv[1:])


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '4082', 'BoardType': 'PXIe', }, }
    example('PXI1Slot2', options, 'DC_VOLTS', 10, 6.5)


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:4082; BoardType:PXIe',]
    _main(cmd_line)


```

(continues on next page)

(continued from previous page)

```
if __name__ == '__main__':
    main()


```

nidmm_multi_point_measurement.py

Listing 3: (nidmm_multi_point_measurement.py)

```
#!/usr/bin/python

import argparse
import nidmm
import sys


def example(resource_name, options, function, range, digits, samples, triggers):
    with nidmm.Session(resource_name=resource_name, options=options) as session:
        session.configure_measurement_digits(measurement_function=nidmm.
↪Function[function], range=range, resolution_digits=digits)
        session.configure_multi_point(trigger_count=triggers, sample_count=samples)
        measurements = session.read_multi_point(array_size=samples)
        print('Measurements: ', measurements)


def _main(argsv):
    parser = argparse.ArgumentParser(description='Performs a multipoint measurement␣
↪using the NI-DMM API.', formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2', help='Resource␣
↪name of an NI digital multimeter.')
    parser.add_argument('-f', '--function', default='DC_VOLTS', choices=nidmm.Function.__
↪members__.keys(), type=str.upper, help='Measurement function.')
    parser.add_argument('-r', '--range', default=10, type=float, help='Measurement range.
↪')
    parser.add_argument('-d', '--digits', default=6.5, type=float, help='Digits of␣
↪resolution for the measurement.')
    parser.add_argument('-s', '--samples', default=10, type=int, help='The number of␣
↪measurements the DMM makes.')
    parser.add_argument('-t', '--triggers', default=1, type=int, help='Sets the number␣
↪of triggers you want the DMM to receive before returning to the Idle state.')
    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.function, args.range, args.
↪digits, args.samples, args.triggers)


def main():
    _main(sys.argv[1:])

```

(continues on next page)

(continued from previous page)

```
32
33 def test_example():
34     options = {'simulate': True, 'driver_setup': {'Model': '4082', 'BoardType': 'PXIe', }
   ↪, }
35     example('PXI1Slot2', options, 'DC_VOLTS', 10, 6.5, 10, 1)
36
37
38 def test_main():
39     cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:4082; BoardType:PXIe',␣
   ↪]
40     _main(cmd_line)
41
42
43 if __name__ == '__main__':
44     main()
45
46
47
```

gRPC Support

Support for using NI-DMM over gRPC

SessionInitializationBehavior

class nidmm.**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 nidmm.**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** (*nidmm.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

n
nidmm, 7

INDEX

A

abort() (*in module nidmm.Session*), 9
AC (*nidmm.WaveformCoupling attribute*), 73
AC_CURRENT (*nidmm.Function attribute*), 67
ac_max_freq (*in module nidmm.Session*), 31
ac_min_freq (*in module nidmm.Session*), 31
AC_VOLTS (*nidmm.Function attribute*), 67
AC_VOLTS_DC_COUPLED (*nidmm.Function attribute*), 68
AcquisitionStatus (*class in nidmm*), 66
adc_calibration (*in module nidmm.Session*), 32
ADCCalibration (*class in nidmm*), 65
aperture_time (*in module nidmm.Session*), 32
aperture_time_units (*in module nidmm.Session*), 33
ApertureTimeUnits (*class in nidmm*), 66
ATTACH_TO_SERVER_SESSION (*nidmm.SessionInitializationBehavior attribute*), 78
AUTO (*nidmm.ADCCalibration attribute*), 65
AUTO (*nidmm.AutoZero attribute*), 66
AUTO (*nidmm.DCNoiseRejection attribute*), 67
AUTO (*nidmm.LCCalculationModel attribute*), 68
AUTO (*nidmm.SessionInitializationBehavior attribute*), 78
auto_range_value (*in module nidmm.Session*), 33
auto_zero (*in module nidmm.Session*), 34
AutoZero (*class in nidmm*), 66
AUX_TRIG1 (*nidmm.SampleTrigger attribute*), 70
AUX_TRIG1 (*nidmm.TriggerSource attribute*), 72

B

B (*nidmm.ThermocoupleType attribute*), 71
buffer_size (*in module nidmm.Session*), 34

C

cable_comp_type (*in module nidmm.Session*), 35
CableCompensationType (*class in nidmm*), 67
CAPACITANCE (*nidmm.Function attribute*), 68
channel_count (*in module nidmm.Session*), 35
close() (*in module nidmm.Session*), 9
configure_measurement_absolute() (*in module nidmm.Session*), 9
configure_measurement_digits() (*in module nidmm.Session*), 11
configure_multi_point() (*in module nidmm.Session*), 12
configure_rtd_custom() (*in module nidmm.Session*), 13
configure_rtd_type() (*in module nidmm.Session*), 13
configure_thermistor_custom() (*in module nidmm.Session*), 15
configure_thermocouple() (*in module nidmm.Session*), 15
configure_trigger() (*in module nidmm.Session*), 16
configure_waveform_acquisition() (*in module nidmm.Session*), 16
current_source (*in module nidmm.Session*), 36
CUSTOM (*nidmm.RTDType attribute*), 69
CUSTOM (*nidmm.ThermistorType attribute*), 71

D

DC (*nidmm.WaveformCoupling attribute*), 73
dc_bias (*in module nidmm.Session*), 36
DC_CURRENT (*nidmm.Function attribute*), 67
dc_noise_rejection (*in module nidmm.Session*), 36
DC_VOLTS (*nidmm.Function attribute*), 67
DCNoiseRejection (*class in nidmm*), 67
DIODE (*nidmm.Function attribute*), 68
disable() (*in module nidmm.Session*), 17
driver_setup (*in module nidmm.Session*), 37
DriverError, 73
DriverNotInstalledError, 73
DriverTooNewError, 74
DriverTooOldError, 73
DriverWarning, 74

E

E (*nidmm.ThermocoupleType attribute*), 71
Error, 73
export_attribute_configuration_buffer() (*in module nidmm.Session*), 17
export_attribute_configuration_file() (*in module nidmm.Session*), 18
EXTERNAL (*nidmm.MeasurementCompleteDest attribute*), 68
EXTERNAL (*nidmm.SampleTrigger attribute*), 70

EXTERNAL (*nidmm.TriggerSource attribute*), 72

F

fetch() (*in module nidmm.Session*), 19
fetch_multi_point() (*in module nidmm.Session*), 19
fetch_waveform() (*in module nidmm.Session*), 20
fetch_waveform_into() (*in module nidmm.Session*), 21
FINISHED_WITH_BACKLOG (*nidmm.AcquisitionStatus attribute*), 66
FINISHED_WITH_NO_BACKLOG (*nidmm.AcquisitionStatus attribute*), 66
FIXED (*nidmm.ThermocoupleReferenceJunctionType attribute*), 71
FOUR_WIRE_RES (*nidmm.Function attribute*), 67
FOUR_WIRE_RTD (*nidmm.TransducerType attribute*), 72
FREQ (*nidmm.Function attribute*), 68
freq_voltage_auto_range (*in module nidmm.Session*), 37
freq_voltage_range (*in module nidmm.Session*), 38
Function (*class in nidmm*), 67
function (*in module nidmm.Session*), 38

G

get_cal_date_and_time() (*in module nidmm.Session*), 21
get_dev_temp() (*in module nidmm.Session*), 22
get_ext_cal_recommended_interval() (*in module nidmm.Session*), 22
get_last_cal_temp() (*in module nidmm.Session*), 22
get_self_cal_supported() (*in module nidmm.Session*), 23
GrpcSessionOptions (*class in nidmm*), 79

H

HIGH_ORDER (*nidmm.DCNoiseRejection attribute*), 67

I

IMMEDIATE (*nidmm.SampleTrigger attribute*), 70
IMMEDIATE (*nidmm.TriggerSource attribute*), 72
import_attribute_configuration_buffer() (*in module nidmm.Session*), 23
import_attribute_configuration_file() (*in module nidmm.Session*), 24
INDUCTANCE (*nidmm.Function attribute*), 68
INITIALIZE_SERVER_SESSION (*nidmm.SessionInitializationBehavior attribute*), 78
initiate() (*in module nidmm.Session*), 25
input_resistance (*in module nidmm.Session*), 39
instrument_firmware_revision (*in module nidmm.Session*), 39
instrument_manufacturer (*in module nidmm.Session*), 39
instrument_model (*in module nidmm.Session*), 40
instrument_product_id (*in module nidmm.Session*), 40
INTERVAL (*nidmm.SampleTrigger attribute*), 70
InvalidRepeatedCapabilityError, 74
io_resource_descriptor (*in module nidmm.Session*), 41
IVIDMM (*nidmm.OperationMode attribute*), 69

J

J (*nidmm.ThermocoupleType attribute*), 71

K

K (*nidmm.ThermocoupleType attribute*), 71

L

LBR_TRIG0 (*nidmm.MeasurementCompleteDest attribute*), 69
LBR_TRIG1 (*nidmm.SampleTrigger attribute*), 70
LBR_TRIG1 (*nidmm.TriggerSource attribute*), 73
lc_calculation_model (*in module nidmm.Session*), 41
lc_number_meas_to_average (*in module nidmm.Session*), 41
LCCalculationModel (*class in nidmm*), 68
lock() (*in module nidmm.Session*), 25
logical_name (*in module nidmm.Session*), 42

M

meas_complete_dest (*in module nidmm.Session*), 42
MeasurementCompleteDest (*class in nidmm*), 68
module
    nidmm, 7

N

N (*nidmm.ThermocoupleType attribute*), 71
nidmm
    module, 7
NO_ACQUISITION_IN_PROGRESS (*nidmm.AcquisitionStatus attribute*), 66
NONE (*nidmm.CableCompensationType attribute*), 67
NONE (*nidmm.MeasurementCompleteDest attribute*), 68
NORMAL (*nidmm.DCNoiseRejection attribute*), 67
number_of_averages (*in module nidmm.Session*), 43

O

OFF (*nidmm.ADCCalibration attribute*), 65
OFF (*nidmm.AutoZero attribute*), 66
offset_comp_ohms (*in module nidmm.Session*), 43
ON (*nidmm.ADCCalibration attribute*), 65
ON (*nidmm.AutoZero attribute*), 66
ONCE (*nidmm.AutoZero attribute*), 66
OPEN (*nidmm.CableCompensationType attribute*), 67

OPEN_AND_SHORT (*nidmm.CableCompensationType attribute*), 67
open_cable_comp_conductance (*in module nidmm.Session*), 44
open_cable_comp_susceptance (*in module nidmm.Session*), 44
operation_mode (*in module nidmm.Session*), 45
OperationMode (*class in nidmm*), 69

P

PARALLEL (*nidmm.LCCalculationModel attribute*), 68
PAUSED (*nidmm.AcquisitionStatus attribute*), 66
perform_open_cable_comp() (*in module nidmm.Session*), 26
perform_short_cable_comp() (*in module nidmm.Session*), 26
PERIOD (*nidmm.Function attribute*), 68
POWER_LINE_CYCLES (*nidmm.ApertureTimeUnits attribute*), 66
powerline_freq (*in module nidmm.Session*), 45
PT3750 (*nidmm.RTDType attribute*), 69
PT3851 (*nidmm.RTDType attribute*), 69
PT3911 (*nidmm.RTDType attribute*), 69
PT3916 (*nidmm.RTDType attribute*), 69
PT3920 (*nidmm.RTDType attribute*), 69
PT3928 (*nidmm.RTDType attribute*), 69
PXI_STAR (*nidmm.SampleTrigger attribute*), 70
PXI_STAR (*nidmm.TriggerSource attribute*), 72
PXI_TRIG0 (*nidmm.MeasurementCompleteDest attribute*), 68
PXI_TRIG0 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG0 (*nidmm.TriggerSource attribute*), 72
PXI_TRIG1 (*nidmm.MeasurementCompleteDest attribute*), 68
PXI_TRIG1 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG1 (*nidmm.TriggerSource attribute*), 72
PXI_TRIG2 (*nidmm.MeasurementCompleteDest attribute*), 69
PXI_TRIG2 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG2 (*nidmm.TriggerSource attribute*), 72
PXI_TRIG3 (*nidmm.MeasurementCompleteDest attribute*), 69
PXI_TRIG3 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG3 (*nidmm.TriggerSource attribute*), 72
PXI_TRIG4 (*nidmm.MeasurementCompleteDest attribute*), 69
PXI_TRIG4 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG4 (*nidmm.TriggerSource attribute*), 72
PXI_TRIG5 (*nidmm.MeasurementCompleteDest attribute*), 69
PXI_TRIG5 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG5 (*nidmm.TriggerSource attribute*), 72
PXI_TRIG6 (*nidmm.MeasurementCompleteDest attribute*), 69
PXI_TRIG6 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG6 (*nidmm.TriggerSource attribute*), 72
PXI_TRIG7 (*nidmm.MeasurementCompleteDest attribute*), 69
PXI_TRIG7 (*nidmm.SampleTrigger attribute*), 70
PXI_TRIG7 (*nidmm.TriggerSource attribute*), 72

R

R (*nidmm.ThermocoupleType attribute*), 71
range (*in module nidmm.Session*), 46
read() (*in module nidmm.Session*), 27
read_multi_point() (*in module nidmm.Session*), 27
read_status() (*in module nidmm.Session*), 28
read_waveform() (*in module nidmm.Session*), 29
reset() (*in module nidmm.Session*), 29
reset_with_defaults() (*in module nidmm.Session*), 30
resolution_absolute (*in module nidmm.Session*), 46
resolution_digits (*in module nidmm.Session*), 47
RpcError, 74
RTDType (*class in nidmm*), 69
RUNNING (*nidmm.AcquisitionStatus attribute*), 66

S

S (*nidmm.ThermocoupleType attribute*), 71
sample_count (*in module nidmm.Session*), 47
sample_interval (*in module nidmm.Session*), 48
sample_trigger (*in module nidmm.Session*), 48
SampleTrigger (*class in nidmm*), 70
SECOND_ORDER (*nidmm.DCNoiseRejection attribute*), 67
SECONDS (*nidmm.ApertureTimeUnits attribute*), 66
self_cal() (*in module nidmm.Session*), 30
self_test() (*in module nidmm.Session*), 30
SelfTestError, 74
send_software_trigger() (*in module nidmm.Session*), 30
serial_number (*in module nidmm.Session*), 49
SERIES (*nidmm.LCCalculationModel attribute*), 68
Session (*class in nidmm*), 7
SessionInitializationBehavior (*class in nidmm*), 78
settle_time (*in module nidmm.Session*), 49
SHORT (*nidmm.CableCompensationType attribute*), 67
short_cable_comp_reactance (*in module nidmm.Session*), 50
short_cable_comp_resistance (*in module nidmm.Session*), 50
simulate (*in module nidmm.Session*), 51
SOFTWARE_TRIG (*nidmm.SampleTrigger attribute*), 70
SOFTWARE_TRIG (*nidmm.TriggerSource attribute*), 72
specific_driver_description (*in module nidmm.Session*), 51
specific_driver_major_version (*in module nidmm.Session*), 52

specific_driver_minor_version (*in module nidmm.Session*), 52
specific_driver_revision (*in module nidmm.Session*), 52
specific_driver_vendor (*in module nidmm.Session*), 53
supported_instrument_models (*in module nidmm.Session*), 53

T

T (*nidmm.ThermocoupleType attribute*), 71
temp_rtd_a (*in module nidmm.Session*), 54
temp_rtd_b (*in module nidmm.Session*), 54
temp_rtd_c (*in module nidmm.Session*), 54
temp_rtd_res (*in module nidmm.Session*), 55
temp_rtd_type (*in module nidmm.Session*), 55
temp_tc_fixed_ref_junc (*in module nidmm.Session*), 56
temp_tc_ref_junc_type (*in module nidmm.Session*), 56
temp_tc_type (*in module nidmm.Session*), 57
temp_thermistor_a (*in module nidmm.Session*), 57
temp_thermistor_b (*in module nidmm.Session*), 57
temp_thermistor_c (*in module nidmm.Session*), 58
temp_thermistor_type (*in module nidmm.Session*), 58
temp_transducer_type (*in module nidmm.Session*), 59
TEMPERATURE (*nidmm.Function attribute*), 68
THERMISTOR (*nidmm.TransducerType attribute*), 72
THERMISTOR_44004 (*nidmm.ThermistorType attribute*), 71
THERMISTOR_44006 (*nidmm.ThermistorType attribute*), 71
THERMISTOR_44007 (*nidmm.ThermistorType attribute*), 71
ThermistorType (*class in nidmm*), 71
THERMOCOUPLE (*nidmm.TransducerType attribute*), 72
ThermocoupleReferenceJunctionType (*class in nidmm*), 71
ThermocoupleType (*class in nidmm*), 71
TransducerType (*class in nidmm*), 72
trigger_count (*in module nidmm.Session*), 59
trigger_delay (*in module nidmm.Session*), 60
trigger_source (*in module nidmm.Session*), 60
TriggerSource (*class in nidmm*), 72
TWO_WIRE_RES (*nidmm.Function attribute*), 67
TWO_WIRE_RTD (*nidmm.TransducerType attribute*), 72

U

unlock() (*in module nidmm.Session*), 31
UnsupportedConfigurationError, 73

W

WAVEFORM (*nidmm.OperationMode attribute*), 69
waveform_coupling (*in module nidmm.Session*), 61
WAVEFORM_CURRENT (*nidmm.Function attribute*), 68
waveform_points (*in module nidmm.Session*), 61
waveform_rate (*in module nidmm.Session*), 62
WAVEFORM_VOLTAGE (*nidmm.Function attribute*), 68
WaveformCoupling (*class in nidmm*), 73