NI-FGEN Python API Documentation

Release 1.4.7

NI

Dec 15, 2023

DOCUMENTATION

CHAPTER

ONE

ABOUT

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

1.1 Support Policy

nifgen supports all the Operating Systems supported by NI-FGEN.

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 nifgen module

4.1.1 Installation

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

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

```
$ python -m pip install nifgen~=1.4.7
```

4.1.2 Usage

The following is a basic example of using the **nifgen** module to open a session to a Function Generator and generate a sine wave for 5 seconds.

```
import nifgen
import time
with nifgen.Session("Dev1") as session:
    session.output_mode = nifgen.OutputMode.FUNC
    session.configure_standard_waveform(waveform=nifgen.Waveform.SINE, amplitude=1.0,␣
↪frequency=10000000, dc_offset=0.0, start_phase=0.0)
    with session.initiate():
        time.sleep(5)
```

Other usage examples can be found on GitHub.

4.1.3 API Reference

Session

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

Creates and returns a new NI-FGEN session to the specified channel of a waveform generator that is used in all subsequent NI-FGEN method calls.

Parameters

- **resource_name** (*str*) –

  Caution: Traditional NI-DAQ and NI-DAQmx device names are not case-sensitive. However, all IVI names, such as logical names, are case-sensitive. If you use logical names, driver session names, or virtual names in your program, you must ensure that the name you use matches the name in the IVI Configuration Store file exactly, without any variations in the case of the characters.

  Specifies the resource name of the device to initialize.

  For Traditional NI-DAQ devices, the syntax is DAQ::*n*, where *n* is the device number assigned by MAX, as shown in Example 1.

  For NI-DAQmx devices, the syntax is just the device name specified in MAX, as shown in Example 2. Typical default names for NI-DAQmx devices in MAX are Dev1 or PXI1Slot1. You can rename an NI-DAQmx device by right-clicking on the name in MAX and entering a new name.

  An alternate syntax for NI-DAQmx devices consists of DAQ::*NI-DAQmx device name*, as shown in Example 3. This naming convention allows for the use of an NI-DAQmx device in an application that was originally designed for a Traditional NI-DAQ device. For example, if the application expects DAQ::1, you can rename the NI-DAQmx device to 1 in MAX and pass in DAQ::1 for the resource name, as shown in Example 4.

  If you use the DAQ::*n* syntax and an NI-DAQmx device name already exists with that same name, the NI-DAQmx device is matched first.

  You can also pass in the name of an IVI logical name or an IVI virtual name configured with the IVI Configuration utility, as shown in Example 5. A logical name identifies a particular virtual instrument. A virtual name identifies a specific device and specifies the initial settings for the session.

Example #	Device Type	Syntax	Variable
1	Traditional NI-DAQ device	DAQ::*1*	(*1* = device number)
2	NI-DAQmx device	*myDAQmxDevice*	(*myDAQmxDevice* = device name)
3	NI-DAQmx device	DAQ::*myDAQmxD*	(*myDAQmxDevice* = device name)
4	NI-DAQmx device	DAQ::*2*	(*2* = device name)
5	IVI logical name or IVI virtual name	*myLogicalName*	(*myLogicalName* = name)

- **channel_name** (*str*, *list*, *range*, *tuple*) – Specifies the channel that this VI uses.

  Default Value: "0"

- **reset_device** (*bool*) – Specifies whether you want to reset the device during the initialization procedure. True specifies that the device is reset and performs the same method as the `nifgen.Session.Reset()` method.

  ****Defined Values****

  Default Value: False

True	Reset device
False	Do not reset device

- **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** (`nifgen.GrpcSessionOptions`) – MeasurementLink gRPC session options

Methods

abort

nifgen.Session.**abort**()

Aborts any previously initiated signal generation. Call the *nifgen.Session.initiate()* method to cause the signal generator to produce a signal again.

allocate_named_waveform

nifgen.Session.**allocate_named_waveform**(*waveform_name*, *waveform_size*)

Specifies the size of a named waveform up front so that it can be allocated in onboard memory before loading the associated data. Data can then be loaded in smaller blocks with the niFgen Write (Binary16) Waveform methods.

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

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

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

Example: `my_session.allocate_named_waveform()`

Parameters

- **waveform_name** (*str*) – Specifies the name to associate with the allocated waveform.
- **waveform_size** (*int*) – Specifies the size of the waveform to allocate in samples.

  Default Value: "4096"

allocate_waveform

nifgen.Session.**allocate_waveform**(*waveform_size*)

Specifies the size of a waveform so that it can be allocated in onboard memory before loading the associated data. Data can then be loaded in smaller blocks with the Write Binary 16 Waveform methods.

Note: The signal generator must not be in the Generating state when you call this method.

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

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

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

Example: `my_session.allocate_waveform()`

Parameters
waveform_size (*int*) – Specifies, in samples, the size of the waveform to allocate.

Return type
int

Returns
The handle that identifies the new waveform. This handle is used later when referring to this waveform.

clear_arb_memory

`nifgen.Session.clear_arb_memory()`

Removes all previously created arbitrary waveforms, sequences, and scripts from the signal generator memory and invalidates all waveform handles, sequence handles, and waveform names.

Note: The signal generator must not be in the Generating state when you call this method.

clear_arb_sequence

`nifgen.Session.clear_arb_sequence(sequence_handle)`

Removes a previously created arbitrary sequence from the signal generator memory and invalidates the sequence handle.

Note: The signal generator must not be in the Generating state when you call this method.

Parameters
sequence_handle (*int*) – Specifies the handle of the arbitrary sequence that you want the signal generator to remove. You can create an arbitrary sequence using the *nifgen.Session.create_arb_sequence()* or *nifgen.Session.create_advanced_arb_sequence()* method. These methods return a handle that you use to identify the sequence.

Defined Value:
NIFGEN_VAL_ALL_SEQUENCES—Remove all sequences from the signal generator

Default Value: None

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.

clear_freq_list

nifgen.Session.**clear_freq_list**(*frequency_list_handle*)

Removes a previously created frequency list from the signal generator memory and invalidates the frequency list handle.

Note: The signal generator must not be in the Generating state when you call this method.

Parameters

frequency_list_handle (*int*) – Specifies the handle of the frequency list you want the signal generator to remove. You create multiple frequency lists using *nifgen.Session.create_freq_list()*. *nifgen.Session.create_freq_list()* returns a handle that you use to identify each list. Specify a value of -1 to clear all frequency lists.

Defined Value

NIFGEN_VAL_ALL_FLISTS—Remove all frequency lists from the signal generator.

Default Value: None

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.

---

clear_user_standard_waveform

nifgen.Session.**clear_user_standard_waveform**()

Clears the user-defined waveform created by the *nifgen.Session.define_user_standard_waveform()* method.

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

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

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

Example: `my_session.clear_user_standard_waveform()`

---

close

nifgen.Session.**close**()

Performs the following operations:

- Closes the instrument I/O session.
- Destroys the NI-FGEN session and all of its properties.
- Deallocates any memory resources NI-FGEN uses.

Not all signal routes established by calling the `nifgen.Session.ExportSignal()` and `nifgen.Session.RouteSignalOut()` methods are released when the NI-FGEN session is closed. The following table shows what happens to a signal route on your device when you call the `nifgen.Session._close()` method.

Routes To	NI 5401/5411/5431	Other Devices
Front Panel	Remain connected	Remain connected
RTSI/PXI Backplane	Remain connected	Disconnected

Note: After calling `nifgen.Session._close()`, you cannot use NI-FGEN again until you call the `nifgen.Session.init()` or `nifgen.Session.InitWithOptions()` methods.

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

commit

`nifgen.Session.`**`commit`**`()`

Causes a transition to the Committed state. This method verifies property values, reserves the device, and commits the property values to the device. If the property values are all valid, NI-FGEN sets the device hardware configuration to match the session configuration. This method does not support the NI 5401/5404/5411/5431 signal generators.

In the Committed state, you can load waveforms, scripts, and sequences into memory. If any properties are changed, NI-FGEN implicitly transitions back to the Idle state, where you can program all session properties before applying them to the device. This method has no effect if the device is already in the Committed or Generating state and returns a successful status value.

Calling this VI before the niFgen Initiate Generation VI is optional but has the following benefits:

- Routes are committed, so signals are exported or imported.
- Any Reference Clock and external clock circuits are phase-locked.
- A subsequent *nifgen.Session.initiate()* method can run faster because the device is already configured.

configure_arb_sequence

`nifgen.Session.`**`configure_arb_sequence`**`(`*sequence_handle*, *gain*, *offset*`)`

Configures the signal generator properties that affect arbitrary sequence generation. Sets the *nifgen.Session.arb_sequence_handle*, *nifgen.Session.arb_gain*, and *nifgen.Session.arb_offset* properties.

Note: The signal generator must not be in the Generating state when you call this method.

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

call this method on the result.

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

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

Example: `my_session.configure_arb_sequence()`

Parameters

- **sequence_handle** (*int*) – Specifies the handle of the arbitrary sequence that you want the signal generator to produce. NI-FGEN sets the *nifgen.Session.arb_sequence_handle* property to this value. You can create an arbitrary sequence using the *nifgen.Session.create_arb_sequence()* or *nifgen.Session.create_advanced_arb_sequence()* method. These methods return a handle that you use to identify the sequence.

  Default Value: None

- **gain** (*float*) – Specifies the factor by which the signal generator scales the arbitrary waveforms in the sequence. When you create an arbitrary waveform, you must first normalize the data points to a range of –1.00 to +1.00. You can use this parameter to scale the waveform to other ranges. The gain is applied before the offset is added.

  For example, to configure the output signal to range from –2.00 to +2.00 V, set **gain** to 2.00.

  Units: unitless

  Default Value: None

- **offset** (*float*) – Specifies the value the signal generator adds to the arbitrary waveform data. When you create arbitrary waveforms, you must first normalize the data points to a range of –1.00 to +1.00 V. You can use this parameter to shift the range of the arbitrary waveform. NI-FGEN sets the *nifgen.Session.arb_offset* property to this value.

  For example, to configure the output signal to range from 0.00 to 2.00 V instead of –1.00 to 1.00 V, set the offset to 1.00.

  Units: volts

  Default Value: None

configure_arb_waveform

nifgen.Session.**configure_arb_waveform**(*waveform_handle*, *gain*, *offset*)

Configures the properties of the signal generator that affect arbitrary waveform generation. Sets the *nifgen.Session.arb_waveform_handle*, *nifgen.Session.arb_gain*, and *nifgen.Session.arb_offset* properties.

Note: The signal generator must not be in the Generating state when you call this method.

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

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

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

Example: `my_session.configure_arb_waveform()`

Parameters

- **waveform_handle** (*int*) – Specifies the handle of the arbitrary waveform you want the signal generator to produce. NI-FGEN sets the *nifgen.Session.arb_waveform_handle* property to this value. You can create an arbitrary waveform using one of the following niFgen Create Waveform methods:

  – `nifgen.Session.create_waveform()`

  – `nifgen.Session.create_waveform()`

  – *nifgen.Session.create_waveform_from_file_i16()*

  – *nifgen.Session.create_waveform_from_file_f64()*

  These methods return a handle that you use to identify the waveform.

  Default Value: None

- **gain** (*float*) – Specifies the factor by which the signal generator scales the arbitrary waveforms in the sequence. When you create an arbitrary waveform, you must first normalize the data points to a range of –1.00 to +1.00. You can use this parameter to scale the waveform to other ranges. The gain is applied before the offset is added.

  For example, to configure the output signal to range from –2.00 to +2.00 V, set **gain** to 2.00.

  Units: unitless

  Default Value: None

- **offset** (*float*) – Specifies the value the signal generator adds to the arbitrary waveform data. When you create arbitrary waveforms, you must first normalize the data points to a range of –1.00 to +1.00 V. You can use this parameter to shift the range of the arbitrary waveform. NI-FGEN sets the *nifgen.Session.arb_offset* property to this value.

  For example, to configure the output signal to range from 0.00 to 2.00 V instead of –1.00 to 1.00 V, set the offset to 1.00.

  Units: volts

  Default Value: None

configure_freq_list

nifgen.Session.**configure_freq_list**(*frequency_list_handle*, *amplitude*, *dc_offset=0.0*, *start_phase=0.0*)

Configures the properties of the signal generator that affect frequency list generation (the *nifgen.Session.freq_list_handle*, *nifgen.Session.func_amplitude*, *nifgen.Session.func_dc_offset*, and *nifgen.Session.func_start_phase* properties).

Note: The signal generator must not be in the Generating state when you call this method.

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

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

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

Example: `my_session.configure_freq_list()`

Parameters

- **frequency_list_handle** (*int*) – Specifies the handle of the frequency list that you want the signal generator to produce. NI-FGEN sets the *nifgen.Session.freq_list_handle* property to this value. You can create a frequency list using the *nifgen.Session.create_freq_list()* method, which returns a handle that you use to identify the list. **Default Value**: None
- **amplitude** (*float*) – Specifies the amplitude of the standard waveform that you want the signal generator to produce. This value is the amplitude at the output terminal. NI-FGEN sets the *nifgen.Session.func_amplitude* property to this value.

  For example, to produce a waveform ranging from –5.00 V to +5.00 V, set the amplitude to 10.00 V.

  Units: peak-to-peak voltage

  Default Value: None

  Note: This parameter does not affect signal generator behavior when you set the **waveform** parameter of the *nifgen.Session.configure_standard_waveform()* method to *DC*.

- **dc_offset** (*float*) – Specifies the DC offset of the standard waveform that you want the signal generator to produce. The value is the offset from ground to the center of the waveform you specify with the **waveform** parameter, observed at the output terminal. For example, to configure a waveform with an amplitude of 10.00 V to range from 0.00 V to +10.00 V, set the **dcOffset** to 5.00 V. NI-FGEN sets the *nifgen.Session.func_dc_offset* property to this value.

  Units: volts

  Default Value: None

- **start_phase** (*float*) – Specifies the horizontal offset of the standard waveform you want the signal generator to produce. Specify this property in degrees of one waveform cycle. NI-FGEN sets the *nifgen.Session.func_start_phase* property to this value. A start phase of 180 degrees means output generation begins halfway through the waveform. A start phase of 360 degrees offsets the output by an entire waveform cycle, which is identical to a start phase of 0 degrees.

  Units: degrees of one cycle

  Default Value: None degrees

Note: This parameter does not affect signal generator behavior when you set the **waveform** parameter to *DC*.

configure_standard_waveform

nifgen.Session.**configure_standard_waveform**(*waveform*, *amplitude*, *frequency*, *dc_offset=0.0*, *start_phase=0.0*)

Configures the following properties of the signal generator that affect standard waveform generation:

- *nifgen.Session.func_waveform*
- *nifgen.Session.func_amplitude*
- *nifgen.Session.func_dc_offset*
- *nifgen.Session.func_frequency*
- *nifgen.Session.func_start_phase*

Note: You must call the `nifgen.Session.ConfigureOutputMode()` method with the **outputMode** parameter set to *FUNC* before calling this method.

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

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

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

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

Example: `my_session.configure_standard_waveform()`

Parameters

- **waveform** (*nifgen.Waveform*) – Specifies the standard waveform that you want the signal generator to produce. NI-FGEN sets the *nifgen.Session.func_waveform* property to this value.

  Defined Values

  Default Value: *SINE*

SINE	Specifies that the signal generator produces a sinusoid waveform.
SQUAR	Specifies that the signal generator produces a square waveform.
TRIAN	Specifies that the signal generator produces a triangle waveform.
*RAMP_*	Specifies that the signal generator produces a positive ramp waveform.
*RAMP_*	Specifies that the signal generator produces a negative ramp waveform.
DC	Specifies that the signal generator produces a constant voltage.
NOISE	Specifies that the signal generator produces white noise.
USER	Specifies that the signal generator produces a user-defined waveform as defined with the *nifgen.Session.define_user_standard_waveform()* method.

- **amplitude** (*float*) – Specifies the amplitude of the standard waveform that you want the signal generator to produce. This value is the amplitude at the output terminal. NI-FGEN sets the *nifgen.Session.func_amplitude* property to this value.

  For example, to produce a waveform ranging from –5.00 V to +5.00 V, set the amplitude to 10.00 V.

  Units: peak-to-peak voltage

  Default Value: None

  Note: This parameter does not affect signal generator behavior when you set the **waveform** parameter of the *nifgen.Session.configure_standard_waveform()* method to *DC*.

- **frequency** (*float*) –

  Specifies the frequency of the standard waveform that you want the signal generator to produce. NI-FGEN sets the *nifgen.Session.func_frequency* property to this value.

  Units: hertz

  Default Value: None

  Note: This parameter does not affect signal generator behavior when you set the **waveform** parameter of the *nifgen.Session.configure_standard_waveform()* method to *DC*.

- **dc_offset** (*float*) – Specifies the DC offset of the standard waveform that you want the signal generator to produce. The value is the offset from ground to the center of the waveform you specify with the **waveform** parameter, observed at the output terminal. For example, to configure a waveform with an amplitude of 10.00 V to range from 0.00 V to +10.00 V, set the **dcOffset** to 5.00 V. NI-FGEN sets the *nifgen.Session.func_dc_offset* property to this value.

  Units: volts

  Default Value: None

- **start_phase** (*float*) – Specifies the horizontal offset of the standard waveform that you want the signal generator to produce. Specify this parameter in degrees of one waveform cycle. NI-FGEN sets the *nifgen.Session.func_start_phase* property to this value. A start phase of 180 degrees means output generation begins

halfway through the waveform. A start phase of 360 degrees offsets the output by an entire waveform cycle, which is identical to a start phase of 0 degrees.

Units: degrees of one cycle

Default Value: 0.00

Note: This parameter does not affect signal generator behavior when you set the **waveform** parameter to *DC*.

---

create_advanced_arb_sequence

nifgen.Session.**create_advanced_arb_sequence**(*waveform_handles_array*, *loop_counts_array*, *sample_counts_array=None*, *marker_location_array=None*)

Creates an arbitrary sequence from an array of waveform handles and an array of corresponding loop counts. This method returns a handle that identifies the sequence. You pass this handle to the *nifgen.Session.configure_arb_sequence()* method to specify what arbitrary sequence you want the signal generator to produce.

The *nifgen.Session.create_advanced_arb_sequence()* method extends on the *nifgen.Session.create_arb_sequence()* method by adding the ability to set the number of samples in each sequence step and to set marker locations.

An arbitrary sequence consists of multiple waveforms. For each waveform, you specify the number of times the signal generator produces the waveform before proceeding to the next waveform. The number of times to repeat a specific waveform is called the loop count.

Note: The signal generator must not be in the Generating state when you call this method. You must call the `nifgen.Session.ConfigureOutputMode()` method to set the **outputMode** parameter to *SEQ* before calling this method.

Parameters

- **waveform_handles_array** (*list of int*) – Specifies the array of waveform handles from which you want to create a new arbitrary sequence. The array must have at least as many elements as the value that you specify in **sequenceLength**. Each **waveformHandlesArray** element has a corresponding **loopCountsArray** element that indicates how many times that waveform is repeated. You obtain waveform handles when you create arbitrary waveforms with the *nifgen.Session.allocate_waveform()* method or one of the following niFgen CreateWaveform methods:

  – `nifgen.Session.create_waveform()`

  – `nifgen.Session.create_waveform()`

  – *nifgen.Session.create_waveform_from_file_i16()*

  – *nifgen.Session.create_waveform_from_file_f64()*

  Default Value: None

- **loop_counts_array** (*list of int*) – Specifies the array of loop counts you want to use to create a new arbitrary sequence. The array must have at least as many elements as the value that you specify in the **sequenceLength** parameter. Each **loopCountsArray** element corresponds to a **waveformHandlesArray** element and indicates how many times to repeat that waveform. Each element of the **loopCountsArray** must be less than or equal to the maximum number of loop counts that the signal generator allows. You can obtain the maximum loop count from **maximumLoopCount** in the *nifgen.Session.query_arb_seq_capabilities()* method.

  Default Value: None

- **sample_counts_array** (*list of int*) – Specifies the array of sample counts that you want to use to create a new arbitrary sequence. The array must have at least as many elements as the value you specify in the **sequenceLength** parameter. Each **sampleCountsArray** element corresponds to a **waveformHandlesArray** element and indicates the subset, in samples, of the given waveform to generate. Each element of the **sampleCountsArray** must be larger than the minimum waveform size, a multiple of the waveform quantum and no larger than the number of samples in the corresponding waveform. You can obtain these values by calling the *nifgen.Session.query_arb_wfm_capabilities()* method.

  Default Value: None

- **marker_location_array** (*list of int*) – Specifies the array of marker locations to where you want a marker to be generated in the sequence. The array must have at least as many elements as the value you specify in the **sequenceLength** parameter. Each **markerLocationArray** element corresponds to a **waveformHandlesArray** element and indicates where in the waveform a marker is to generate. The marker location must be less than the size of the waveform the marker is in. The markers are coerced to the nearest marker quantum and the coerced values are returned in the **coercedMarkersArray** parameter.

  If you do not want a marker generated for a particular sequence stage, set this parameter to NIFGEN_VAL_NO_MARKER.

  Defined Value: NIFGEN_VAL_NO_MARKER

  Default Value: None

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

tuple (coerced_markers_array, sequence_handle)

WHERE

coerced_markers_array (list of int):

Returns an array of all given markers that are coerced (rounded) to the nearest marker quantum. Not all devices coerce markers.

Default Value: None

sequence_handle (int):

Returns the handle that identifies the new arbitrary sequence. You can pass this handle to *nifgen.Session.configure_arb_sequence()* to generate the arbitrary sequence.

create_arb_sequence

nifgen.Session.**create_arb_sequence**(*waveform_handles_array*, *loop_counts_array*)

Creates an arbitrary sequence from an array of waveform handles and an array of corresponding loop counts. This method returns a handle that identifies the sequence. You pass this handle to the *nifgen.Session.configure_arb_sequence()* method to specify what arbitrary sequence you want the signal generator to produce.

An arbitrary sequence consists of multiple waveforms. For each waveform, you can specify the number of times that the signal generator produces the waveform before proceeding to the next waveform. The number of times to repeat a specific waveform is called the loop count.

Note: You must call the `nifgen.Session.ConfigureOutputMode()` method to set the **outputMode** parameter to *SEQ* before calling this method.

Parameters

- **waveform_handles_array** (*list of int*) – Specifies the array of waveform handles from which you want to create a new arbitrary sequence. The array must have at least as many elements as the value that you specify in **sequenceLength**. Each **waveformHandlesArray** element has a corresponding **loopCountsArray** element that indicates how many times that waveform is repeated. You obtain waveform handles when you create arbitrary waveforms with the *nifgen.Session.allocate_waveform()* method or one of the following niFgen CreateWaveform methods:

  – `nifgen.Session.create_waveform()`

  – `nifgen.Session.create_waveform()`

  – *nifgen.Session.create_waveform_from_file_i16()*

  – *nifgen.Session.create_waveform_from_file_f64()*

  Default Value: None

- **loop_counts_array** (*list of int*) – Specifies the array of loop counts you want to use to create a new arbitrary sequence. The array must have at least as many elements as the value that you specify in the **sequenceLength** parameter. Each **loopCountsArray** element corresponds to a **waveformHandlesArray** element and indicates how many times to repeat that waveform. Each element of the **loopCountsArray** must be less than or equal to the maximum number of loop counts that the signal generator allows. You can obtain the maximum loop count from **maximumLoopCount** in the *nifgen.Session.query_arb_seq_capabilities()* method.

  Default Value: None

Return type
int

Returns
Returns the handle that identifies the new arbitrary sequence. You can pass this handle to *nifgen.Session.configure_arb_sequence()* to generate the arbitrary sequence.

create_freq_list

nifgen.Session.**create_freq_list**(*waveform*, *frequency_array*, *duration_array*)

Creates a frequency list from an array of frequencies (**frequencyArray**) and an array of durations (**durationArray**). The two arrays should have the same number of elements, and this value must also be the size of the **frequencyListLength**. The method returns a handle that identifies the frequency list (the **frequencyListHandle**). You can pass this handle to *nifgen.Session.configure_freq_list()* to specify what frequency list you want the signal generator to produce.

A frequency list consists of a list of frequencies and durations. The signal generator generates each frequency for the given amount of time and then proceeds to the next frequency. When the end of the list is reached, the signal generator starts over at the beginning of the list.

Note: The signal generator must not be in the Generating state when you call this method.

Parameters

- **waveform** (*nifgen.Waveform*) – Specifies the standard waveform that you want the signal generator to produce. NI-FGEN sets the *nifgen.Session.func_waveform* property to this value.

  ****Defined Values****

  Default Value: *SINE*

SINE	Specifies that the signal generator produces a sinusoid waveform.
SQUAR	Specifies that the signal generator produces a square waveform.
TRIAN	Specifies that the signal generator produces a triangle waveform.
*RAMP_*	Specifies that the signal generator produces a positive ramp waveform.
*RAMP_*	Specifies that the signal generator produces a negative ramp waveform.
DC	Specifies that the signal generator produces a constant voltage.
NOISE	Specifies that the signal generator produces white noise.
USER	Specifies that the signal generator produces a user-defined waveform as defined with the *nifgen.Session.define_user_standard_waveform()* method.

- **frequency_array** (*list of float*) – Specifies the array of frequencies to form the frequency list. The array must have at least as many elements as the value you specify in **frequencyListLength**. Each **frequencyArray** element has a corresponding **durationArray** element that indicates how long that frequency is repeated.

  Units: hertz

  Default Value: None

- **duration_array** (*list of float*) – Specifies the array of durations to form the frequency list. The array must have at least as many elements as the value that you specify in **frequencyListLength**. Each **durationArray** element has a corresponding **frequencyArray** element and indicates how long in seconds to generate the corresponding frequency.

  Units: seconds

  Default Value: None

Return type
int

Returns
Returns the handle that identifies the new frequency list. You can pass this handle to *nifgen.Session.configure_freq_list()* to generate the arbitrary sequence.

create_waveform_from_file_f64

nifgen.Session.**create_waveform_from_file_f64**(*file_name*, *byte_order*)

This method takes the floating point double (F64) data from the specified file and creates an onboard waveform for use in Arbitrary Waveform or Arbitrary Sequence output mode. The **waveformHandle** returned by this method can later be used for setting the active waveform, changing the data in the waveform, building sequences of waveforms, or deleting the waveform when it is no longer needed.

Note: The F64 data must be between –1.0 and +1.0 V. Use the *nifgen.Session.digital_gain* property to generate different voltage outputs.

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

Example: `my_session.channels[ ... ].create_waveform_from_file_f64()`

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

Example: `my_session.create_waveform_from_file_f64()`

Parameters

- **file_name** (*str*) – The full path and name of the file where the waveform data resides.
- **byte_order** (*nifgen.ByteOrder*) – Specifies the byte order of the data in the file.

  ****Defined Values****

  ****Default Value**:** *LITTLE*

LITTL	Little Endian Data—The least significant bit is stored at the lowest address, followed by the other bits, in order of increasing significance.
BIG	Big Endian Data—The most significant bit is stored at the lowest address, followed by the other bits, in order of decreasing significance.

Note: Data written by most applications in Windows (including LabWindows™/CVI™) is in Little Endian format. Data written to a file from LabVIEW

is in Big Endian format by default on all platforms. Big Endian and Little Endian refer to the way data is stored in memory, which can differ on different processors.

Return type
int

Returns
The handle that identifies the new waveform. This handle is used later when referring to this waveform.

create_waveform_from_file_i16

nifgen.Session.**create_waveform_from_file_i16**(*file_name*, *byte_order*)

Takes the binary 16-bit signed integer (I16) data from the specified file and creates an onboard waveform for use in Arbitrary Waveform or Arbitrary Sequence output mode. The **waveformHandle** returned by this method can later be used for setting the active waveform, changing the data in the waveform, building sequences of waveforms, or deleting the waveform when it is no longer needed.

Note: The I16 data (values between –32768 and +32767) is assumed to represent –1 to +1 V. Use the *nifgen.Session.digital_gain* property to generate different voltage outputs.

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

Example: `my_session.channels[ ... ].create_waveform_from_file_i16()`

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

Example: `my_session.create_waveform_from_file_i16()`

Parameters

- **file_name** (*str*) – The full path and name of the file where the waveform data resides.
- **byte_order** (*nifgen.ByteOrder*) – Specifies the byte order of the data in the file.

  Defined Values

  Default Value: *LITTLE*

LITTL	Little Endian Data—The least significant bit is stored at the lowest address, followed by the other bits, in order of increasing significance.
BIG	Big Endian Data—The most significant bit is stored at the lowest address, followed by the other bits, in order of decreasing significance.

Note: Data written by most applications in Windows (including LabWindows™/CVI™) is in Little Endian format. Data written to a file from LabVIEW is in Big Endian format by default on all platforms. Big Endian and Little Endian refer to the way data is stored in memory, which can differ on different processors.

Return type
int

Returns
The handle that identifies the new waveform. This handle is used later when referring to this waveform.

create_waveform_numpy

nifgen.Session.**create_waveform_numpy**(*waveform_data_array*)

Creates an onboard waveform for use in Arbitrary Waveform output mode or Arbitrary Sequence output mode.

Note: You must set *nifgen.Session.output_mode* to *ARB* or *SEQ* before calling this method.

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

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

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

Example: `my_session.create_waveform()`

Parameters
waveform_data_array (*iterable of float or int16*) – Array of data for the new arbitrary waveform. This may be an iterable of float or int16, or for best performance a numpy.ndarray of dtype int16 or float64.

Return type
int

Returns
The handle that identifies the new waveform. This handle is used in other methods when referring to this waveform.

define_user_standard_waveform

nifgen.Session.**define_user_standard_waveform**(*waveform_data_array*)

Defines a user waveform for use in either Standard Method or Frequency List output mode.

To select the waveform, set the **waveform** parameter to *USER* with either the *nifgen.Session.configure_standard_waveform()* or the *nifgen.Session.create_freq_list()* method.

The waveform data must be scaled between –1.0 and 1.0. Use the **amplitude** parameter in the *nifgen.Session.configure_standard_waveform()* method to generate different output voltages.

Note: You must call the `nifgen.Session.ConfigureOutputMode()` method to set the **outputMode** parameter to *FUNC* or *FREQ_LIST* before calling this method.

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

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

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

Example: `my_session.define_user_standard_waveform()`

Parameters

waveform_data_array (*list of float*) – Specifies the array of data you want to use for the new arbitrary waveform. The array must have at least as many elements as the value that you specify in **waveformSize**.

You must normalize the data points in the array to be between –1.00 and +1.00.

Default Value: None

delete_script

nifgen.Session.**delete_script**(*script_name*)

Deletes the specified script from onboard memory.

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

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

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

Example: `my_session.delete_script()`

Parameters

script_name (*str*) – Specifies the name of the script you want to delete. The script name appears in the text of the script following the script keyword.

delete_waveform

nifgen.Session.**delete_waveform**(*waveform_name_or_handle*)

Removes a previously created arbitrary waveform from the signal generator memory.

Note: The signal generator must not be in the Generating state when you call this method.

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

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

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

Example: `my_session.delete_waveform()`

Parameters

waveform_name_or_handle (*str or int*) – The name (str) or handle (int) of an arbitrary waveform previously allocated with *nifgen.Session.allocate_named_waveform()*, *nifgen.Session.allocate_waveform()* or nifgen.Session.create_waveform().

disable

nifgen.Session.**disable**()

Places the instrument in a quiescent state where it has minimal or no impact on the system to which it is connected. The analog output and all exported signals are disabled.

export_attribute_configuration_buffer

nifgen.Session.**export_attribute_configuration_buffer**()

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

You can export and import session property configurations only between devices with identical model numbers, channel counts, and onboard memory sizes.

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

Return type

bytes

Returns

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

export_attribute_configuration_file

nifgen.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, channel counts, and onboard memory sizes.

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

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:** .nifgenconfig

get_channel_name

nifgen.Session.**get_channel_name**(*index*)

Returns the channel string that is in the channel table at an index you specify.

Note: This method is included for compliance with the IviFgen Class Specification.

Parameters

index (*int*) – A 1-based index into the channel table.

Return type

str

Returns

Returns the channel string that is in the channel table at the index you specify. Do not modify the contents of the channel string.

get_ext_cal_last_date_and_time

nifgen.Session.**get_ext_cal_last_date_and_time**()

Returns the date and time of the last successful external calibration. The time returned is 24-hour (military) local time; for example, if the device was calibrated at 2:30 PM, this method returns 14 for the **hour** parameter and 30 for the **minute** parameter.

Return type

hightime.datetime

Returns

Indicates date and time of the last calibration.

get_ext_cal_last_temp

nifgen.Session.**get_ext_cal_last_temp**()

Returns the temperature at the last successful external calibration. The temperature is returned in degrees Celsius.

Return type
float

Returns
Specifies the temperature at the last successful calibration in degrees Celsius.

get_ext_cal_recommended_interval

nifgen.Session.**get_ext_cal_recommended_interval**()

Returns the recommended interval between external calibrations in months.

Return type
hightime.timedelta

Returns
Specifies the recommended interval between external calibrations in months.

get_hardware_state

nifgen.Session.**get_hardware_state**()

Returns the current hardware state of the device and, if the device is in the hardware error state, the current hardware error.

Note: Hardware states do not necessarily correspond to NI-FGEN states.

Return type
nifgen.HardwareState

Returns

Returns the hardware state of the signal generator.

Defined Values

IDLE	The device is in the Idle state.
WAITING_FOR_START_TRIGGER	The device is waiting for Start Trigger.
RUNNING	The device is in the Running state.
DONE	The generation has completed successfully.
HARDWARE_ERROR	There is a hardware error.

get_self_cal_last_date_and_time

nifgen.Session.**get_self_cal_last_date_and_time**()

Returns the date and time of the last successful self-calibration.

Return type
hightime.datetime

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

get_self_cal_last_temp

nifgen.Session.**get_self_cal_last_temp**()

Returns the temperature at the last successful self-calibration. The temperature is returned in degrees Celsius.

Return type
float

Returns
Specifies the temperature at the last successful calibration in degrees Celsius.

get_self_cal_supported

nifgen.Session.**get_self_cal_supported**()

Returns whether the device supports self–calibration.

Return type
bool

Returns

Returns whether the device supports self-calibration.

****Defined Values****

True	Self–calibration is supported.
False	Self–calibration is not supported.

import_attribute_configuration_buffer

nifgen.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, channel counts, and onboard memory sizes.

Note: You cannot call this method while the session is in a running state, such as while generating a signal.

Parameters

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

import_attribute_configuration_file

nifgen.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, channel counts, and onboard memory sizes.

Note: You cannot call this method while the session is in a running state, such as while generating a signal.

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:** .nifgenconfig

initiate

nifgen.Session.**initiate**()

Initiates signal generation. If you want to abort signal generation, call the *nifgen.Session.abort()* method. After the signal generation is aborted, you can call the *nifgen.Session.initiate()* method to cause the signal generator to produce a signal again.

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

---

is_done

nifgen.Session.**is_done**()

Determines whether the current generation is complete. This method sets the **done** parameter to True if the session is in the Idle or Committed states.

Note: NI-FGEN only reports the **done** parameter as True after the current generation is complete in Single trigger mode.

Return type

bool

Returns

Returns information about the completion of waveform generation.

Defined Values

True	Generation is complete.
False	Generation is not complete.

lock

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

query_arb_seq_capabilities

nifgen.Session.**query_arb_seq_capabilities**()

Returns the properties of the signal generator that are related to creating arbitrary sequences (the *nifgen.Session.max_num_sequences*, *nifgen.Session.min_sequence_length*, *nifgen.Session.max_sequence_length*, and *nifgen.Session.max_loop_count* properties).

Return type

tuple (maximum_number_of_sequences, minimum_sequence_length, maximum_sequence_length, maximum_loop_count)

WHERE

maximum_number_of_sequences (int):

Returns the maximum number of arbitrary waveform sequences that the signal generator allows. NI-FGEN obtains this value from the *nifgen.Session.max_num_sequences* property.

minimum_sequence_length (int):

Returns the minimum number of arbitrary waveforms the signal generator allows in a sequence. NI-FGEN obtains this value from the *nifgen.Session.min_sequence_length* property.

maximum_sequence_length (int):

Returns the maximum number of arbitrary waveforms the signal generator allows in a sequence. NI-FGEN obtains this value from the *nifgen.Session.max_sequence_length* property.

maximum_loop_count (int):

Returns the maximum number of times the signal generator can repeat an arbitrary waveform in a sequence. NI-FGEN obtains this value from the *nifgen.Session.max_loop_count* property.

query_arb_wfm_capabilities

nifgen.Session.**query_arb_wfm_capabilities**()

Returns the properties of the signal generator that are related to creating arbitrary waveforms. These properties are the maximum number of waveforms, waveform quantum, minimum waveform size, and maximum waveform size.

Note: If you do not want to obtain the waveform quantum, pass a value of VI_NULL for this parameter.

Return type

tuple (maximum_number_of_waveforms, waveform_quantum, minimum_waveform_size, maximum_waveform_size)

WHERE

maximum_number_of_waveforms (int):

Returns the maximum number of arbitrary waveforms that the signal generator allows. NI-FGEN obtains this value from the *nifgen.Session.max_num_waveforms* property.

waveform_quantum (int):

The size (number of points) of each waveform must be a multiple of a constant quantum value. This parameter obtains the quantum value that the signal generator uses. NI-FGEN returns this value from the *nifgen.Session.waveform_quantum* property.

For example, when this property returns a value of 8, all waveform sizes must be a multiple of 8.

minimum_waveform_size (int):

Returns the minimum number of points that the signal generator allows in a waveform. NI-FGEN obtains this value from the *nifgen.Session.min_waveform_size* property.

maximum_waveform_size (int):

Returns the maximum number of points that the signal generator allows in a waveform. NI-FGEN obtains this value from the *nifgen.Session.max_waveform_size* property.

query_freq_list_capabilities

nifgen.Session.**query_freq_list_capabilities**()

Returns the properties of the signal generator that are related to creating frequency lists. These properties are *nifgen.Session.max_num_freq_lists*, *nifgen.Session.min_freq_list_length*, *nifgen.Session.max_freq_list_length*, *nifgen.Session.min_freq_list_duration*, *nifgen.Session.max_freq_list_duration*, and *nifgen.Session.freq_list_duration_quantum*.

Return type

tuple (maximum_number_of_freq_lists, minimum_frequency_list_length, maximum_frequency_list_length, minimum_frequency_list_duration, maximum_frequency_list_duration, frequency_list_duration_quantum)

WHERE

maximum_number_of_freq_lists (int):

Returns the maximum number of frequency lists that the signal generator allows. NI-FGEN obtains this value from the *nifgen.Session.max_num_freq_lists* property.

minimum_frequency_list_length (int):

Returns the minimum number of steps that the signal generator allows in a frequency list. NI-FGEN obtains this value from the *nifgen.Session.min_freq_list_length* property.

maximum_frequency_list_length (int):

Returns the maximum number of steps that the signal generator allows in a frequency list. NI-FGEN obtains this value from the *nifgen.Session.max_freq_list_length* property.

minimum_frequency_list_duration (float):

Returns the minimum duration that the signal generator allows in a step of a frequency list. NI-FGEN obtains this value from the *nifgen.Session.min_freq_list_duration* property.

maximum_frequency_list_duration (float):

Returns the maximum duration that the signal generator allows in a step of a frequency list. NI-FGEN obtains this value from the *nifgen.Session.max_freq_list_duration* property.

frequency_list_duration_quantum (float):

Returns the quantum of which all durations must be a multiple in a frequency list. NI-FGEN obtains this value from the *nifgen.Session.freq_list_duration_quantum* property.

read_current_temperature

nifgen.Session.**read_current_temperature**()

Reads the current onboard temperature of the device. The temperature is returned in degrees Celsius.

Return type
float

Returns
Returns the current temperature read from onboard temperature sensors, in degrees Celsius.

reset

nifgen.Session.**reset**()

Resets the instrument to a known state. This method aborts the generation, clears all routes, and resets session properties to the default values. This method does not, however, commit the session properties or configure the device hardware to its default state.

Note: For the NI 5401/5404/5411/5431, this method exhibits the same behavior as the *nifgen.Session.reset_device()* method.

---

reset_device

nifgen.Session.**reset_device**()

Performs a hard reset on the device. Generation is stopped, all routes are released, external bidirectional terminals are tristated, FPGAs are reset, hardware is configured to its default state, and all session properties are reset to their default states.

reset_with_defaults

nifgen.Session.**reset_with_defaults**()

Resets the instrument and reapplies initial user–specified settings from the logical name that was used to initialize the session. If the session was created without a logical name, this method is equivalent to the *nifgen.Session.reset()* method.

self_cal

nifgen.Session.**self_cal**()

Performs a full internal self-calibration on the device. If the calibration is successful, new calibration data and constants are stored in the onboard EEPROM.

self_test

nifgen.Session.**self_test**()

Runs the instrument self-test routine and returns the test result(s).

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

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

Self-Test Code	Description
0	Passed self-test
1	Self-test failed

Note: When used on some signal generators, the device is reset after the *nifgen.Session.self_test()* method runs. If you use the *nifgen.Session.self_test()* method, your device may not be in its previously configured state after the method runs.

send_software_edge_trigger

nifgen.Session.**send_software_edge_trigger**(*trigger*, *trigger_id*)

Sends a command to trigger the signal generator. This VI can act as an override for an external edge trigger.

Note: This VI does not override external digital edge triggers of the NI 5401/5411/5431.

Parameters

- **trigger** (*nifgen.Trigger*) – Trigger specifies the type of software trigger to send

Defined Values
START
SCRIPT

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.

- **trigger_id** (*str*) – Trigger ID specifies the Script Trigger to use for triggering.

set_next_write_position

nifgen.Session.**set_next_write_position**(*waveform_name_or_handle*, *relative_to*, *offset*)

Sets the position in the waveform at which the next waveform data is written. This method allows you to write to arbitrary locations within the waveform. These settings apply only to the next write to the waveform specified by the waveformHandle parameter. Subsequent writes to that waveform begin where the last write left off, unless this method is called again. The waveformHandle passed in must have been created by a call to the *nifgen.Session.allocate_waveform()* method or one of the following nifgen.Session.create_waveform() method.

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

Example: my_session.channels[...].set_next_write_position()

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

Example: my_session.set_next_write_position()

Parameters

- **waveform_name_or_handle** (*str or int*) – The name (str) or handle (int) of an arbitrary waveform previously allocated with *nifgen.Session.allocate_named_waveform()*, *nifgen.Session.allocate_waveform()* or nifgen.Session.create_waveform().
- **relative_to** (*nifgen.RelativeTo*) – Specifies the reference position in the waveform. This position and **offset** together determine where to start loading data into the waveform.

  Defined Values

START (0)	Use the start of the waveform as the reference position.
CURRENT (1)	Use the current position within the waveform as the reference position.

- **offset** (*int*) – Specifies the offset from **relativeTo** at which to start loading the data into the waveform.

unlock

nifgen.Session.**unlock**()

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

wait_until_done

nifgen.Session.**wait_until_done**(*max_time=hightime.timedelta(seconds=10.0)*)

Waits until the device is done generating or until the maximum time has expired.

Parameters

max_time (*hightime.timedelta, datetime.timedelta, or int in milliseconds*) – Specifies the timeout value in milliseconds.

write_script

nifgen.Session.**write_script**(*script*)

Writes a string containing one or more scripts that govern the generation of waveforms.

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

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

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

Example: `my_session.write_script()`

Parameters

script (*str*) – Contains the text of the script you want to use for your generation operation. Refer to scripting Instructions for more information about writing scripts.

write_waveform

nifgen.Session.**write_waveform**(*waveform_name_or_handle*, *data*)

Writes data to the waveform in onboard memory.

By default, subsequent calls to this method continue writing data from the position of the last sample written. You can set the write position and offset by calling the *nifgen.Session.set_next_write_position()* *nifgen.Session.set_next_write_position()* method.

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

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

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

Example: `my_session.write_waveform()`

Parameters

- **waveform_name_or_handle** (*str or int*) – The name (str) or handle (int) of an arbitrary waveform previously allocated with *nifgen.Session.allocate_named_waveform()*, *nifgen.Session.allocate_waveform()* or `nifgen.Session.create_waveform()`.

- **data** (*list of float*) – Array of data to load into the waveform. This may be an iterable of float, or for best performance a numpy.ndarray of dtype int16 or float64.

Properties

absolute_delay

nifgen.Session.**absolute_delay**

Specifies the sub-Sample Clock delay, in seconds, to apply to the waveform. Use this property to reduce the trigger jitter when synchronizing multiple devices with NI-TClk. This property can also help maintain synchronization repeatability by writing the absolute delay value of a previous measurement to the current session. To set this property, the waveform generator must be in the Idle (Configuration) state. **Units**: seconds (s) **Valid Values**: Plus or minus half of one Sample Clock period **Default Value**: 0.0 **Supported Waveform Generators**: PXIe-5413/5423/5433

Note: If this property is set, NI-TClk cannot perform any sub-Sample Clock adjustment.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Absolute Delay**
- C Attribute: **NIFGEN_ATTR_ABSOLUTE_DELAY**

---

all_marker_events_latched_status

nifgen.Session.**all_marker_events_latched_status**

Returns a bit field of the latched status of all Marker Events. Write 0 to this property to clear the latched status of all Marker Events.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Events:Marker:Advanced:All Marker Events Latched Status**

- C Attribute: **NIFGEN_ATTR_ALL_MARKER_EVENTS_LATCHED_STATUS**

all_marker_events_live_status

nifgen.Session.**all_marker_events_live_status**

Returns a bit field of the live status of all Marker Events.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Events:Marker:Advanced:All Marker Events Live Status**
- C Attribute: **NIFGEN_ATTR_ALL_MARKER_EVENTS_LIVE_STATUS**

analog_data_mask

nifgen.Session.**analog_data_mask**

Specifies the mask to apply to the analog output. The masked data is replaced with the data in *nifgen.Session.analog_static_value*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Data Mask:Analog Data Mask**
- C Attribute: **NIFGEN_ATTR_ANALOG_DATA_MASK**

analog_filter_enabled

nifgen.Session.**analog_filter_enabled**

Controls whether the signal generator applies to an analog filter to the output signal. This property is valid in arbitrary waveform, arbitrary sequence, and script modes. This property can also be used in standard method and frequency list modes for user-defined waveforms.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Filters:Analog Filter Enabled**
- C Attribute: **NIFGEN_ATTR_ANALOG_FILTER_ENABLED**

analog_path

nifgen.Session.**analog_path**

Specifies the analog signal path that should be used. The main path allows you to configure gain, offset, analog filter status, output impedance, and output enable. The main path has two amplifier options, high- and low-gain. The direct path presents a much smaller gain range, and you cannot adjust offset or the filter status. The direct path also provides a smaller output range but also lower distortion. NI-FGEN normally chooses the amplifier based on the user-specified gain.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.AnalogPath
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Analog Path**
- C Attribute: **NIFGEN_ATTR_ANALOG_PATH**

analog_static_value

nifgen.Session.**analog_static_value**

Specifies the static value that replaces data masked by *nifgen.Session.analog_data_mask*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Data Mask:Analog Static Value**
- C Attribute: **NIFGEN_ATTR_ANALOG_STATIC_VALUE**

arb_gain

nifgen.Session.**arb_gain**

Specifies the factor by which the signal generator scales the arbitrary waveform data. When you create arbitrary waveforms, you must first normalize the data points to the range -1.0 to +1.0. Use this property to scale the arbitrary waveform to other ranges. For example, when you set this property to 2.0, the output signal ranges from -2.0 V to +2.0 V. Use this property when *nifgen.Session.output_mode* is set to *ARB* or *SEQ*.

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

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

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

Example: `my_session.arb_gain`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Arbitrary Waveform:Gain**
- C Attribute: **NIFGEN_ATTR_ARB_GAIN**

arb_marker_position

nifgen.Session.**arb_marker_position**

Specifies the position for a marker to be asserted in the arbitrary waveform. This property defaults to -1 when no marker position is specified. Use this property when *nifgen.Session.output_mode* is set to *ARB*. Use `nifgen.Session.ExportSignal()` to export the marker signal.

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

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Waveform Mode:Marker Position**
- C Attribute: **NIFGEN_ATTR_ARB_MARKER_POSITION**

arb_offset

nifgen.Session.**arb_offset**

Specifies the value that the signal generator adds to the arbitrary waveform data. When you create arbitrary waveforms, you must first normalize the data points to the range -1.0 to +1.0. Use this property to shift the arbitrary waveform range. For example, when you set this property to 1.0, the output signal ranges from 2.0 V to 0.0 V. Use this property when *nifgen.Session.output_mode* is set to *ARB* or *SEQ*. Units: Volts

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

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

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

Example: `my_session.arb_offset`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Arbitrary Waveform:Offset**
- C Attribute: **NIFGEN_ATTR_ARB_OFFSET**

arb_repeat_count

nifgen.Session.**arb_repeat_count**

Specifies number of times to repeat the arbitrary waveform when the triggerMode parameter of `nifgen.Session.ConfigureTriggerMode()` is set to *SINGLE* or *STEPPED*. This property is ignored if the triggerMode parameter is set to *CONTINUOUS* or *BURST*. Use this property when *nifgen.Session.output_mode* is set to *ARB*. When used during streaming, this property specifies the number of times to repeat the streaming waveform (the onboard memory allocated for streaming). For more information about streaming, refer to the Streaming topic.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Waveform Mode:Repeat Count**
- C Attribute: **NIFGEN_ATTR_ARB_REPEAT_COUNT**

arb_sample_rate

nifgen.Session.**arb_sample_rate**

Specifies the rate at which the signal generator outputs the points in arbitrary waveforms. Use this property when *nifgen.Session.output_mode* is set to *ARB* or *SEQ*. Units: Samples/s

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock:Rate**
- C Attribute: **NIFGEN_ATTR_ARB_SAMPLE_RATE**

arb_sequence_handle

nifgen.Session.**arb_sequence_handle**

This channel-based property identifies which sequence the signal generator produces. You can create multiple sequences using *nifgen.Session.create_arb_sequence()*. *nifgen.Session.create_arb_sequence()* returns a handle that you can use to identify the particular sequence. To configure the signal generator to produce a particular sequence, set this property to the sequence handle. Use this property only when *nifgen.Session.output_mode* is set to *SEQ*.

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

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

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

Example: `my_session.arb_sequence_handle`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Sequence Mode:Arbitrary Sequence Handle**
- C Attribute: **NIFGEN_ATTR_ARB_SEQUENCE_HANDLE**

---

arb_waveform_handle

nifgen.Session.**arb_waveform_handle**

Selects which arbitrary waveform the signal generator produces. You can create multiple arbitrary waveforms using one of the following niFgen Create Waveform methods: `nifgen.Session.create_waveform()` `nifgen.Session.create_waveform()` *nifgen.Session.create_waveform_from_file_i16()* *nifgen.Session.create_waveform_from_file_f64()* These methods return a handle that you can use to identify the particular waveform. To configure the signal generator to produce a particular waveform, set this property to the waveform handle. Use this property only when *nifgen.Session.output_mode* is set to *ARB*.

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

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

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

Example: `my_session.arb_waveform_handle`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Waveform Mode:Arbitrary Waveform Handle**
- C Attribute: **NIFGEN_ATTR_ARB_WAVEFORM_HANDLE**

aux_power_enabled

`nifgen.Session.`**`aux_power_enabled`**

Controls the specified auxiliary power pin. Setting this property to TRUE energizes the auxiliary power when the session is committed. When this property is FALSE, the power pin of the connector outputs no power.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Advanced:AUX Power Enabled**
- C Attribute: **NIFGEN_ATTR_AUX_POWER_ENABLED**

bus_type

`nifgen.Session.`**`bus_type`**

The bus type of the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.BusType
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Bus Type**
- C Attribute: **NIFGEN_ATTR_BUS_TYPE**

channel_delay

nifgen.Session.**channel_delay**

Specifies, in seconds, the delay to apply to the analog output of the channel specified by the channel string. You can use the channel delay to configure the timing relationship between channels on a multichannel device. Values for this property can be zero or positive. A value of zero indicates that the channels are aligned. A positive value delays the analog output by the specified number of seconds.

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

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

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

Example: `my_session.channel_delay`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Output:Channel Delay**
- C Attribute: **NIFGEN_ATTR_CHANNEL_DELAY**

clock_mode

nifgen.Session.**clock_mode**

Controls which clock mode is used for the signal generator. For signal generators that support it, this property allows switching the sample clock to High-Resolution mode. When in Divide-Down mode, the sample rate can only be set to certain frequences, based on dividing down the update clock. However, in High-Resolution mode, the sample rate may be set to any value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ClockMode
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock:Mode**
- C Attribute: **NIFGEN_ATTR_CLOCK_MODE**

common_mode_offset

`nifgen.Session.common_mode_offset`

Specifies, in volts, the value the signal generator adds to or subtracts from the arbitrary waveform data. This property applies only when you set the *nifgen.Session.terminal_configuration* property to *DIFFERENTIAL*. Common mode offset is applied to the signals generated at each differential output terminal.

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

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

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

Example: `my_session.common_mode_offset`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Output:Common Mode Offset**
- C Attribute: **NIFGEN_ATTR_COMMON_MODE_OFFSET**

data_marker_events_count

nifgen.Session.**data_marker_events_count**

Returns the number of Data Marker Events supported by the device.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Data Marker Events Count**
- C Attribute: **NIFGEN_ATTR_DATA_MARKER_EVENTS_COUNT**

data_marker_event_data_bit_number

nifgen.Session.**data_marker_event_data_bit_number**

Specifies the bit number to assign to the Data Marker Event.

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

Example: `my_session.data_markers[ ... ].data_marker_event_data_bit_number`

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

Example: `my_session.data_marker_event_data_bit_number`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	data_markers

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

- LabVIEW Property: **Events:Data Marker:Data Bit Number**
- C Attribute: **NIFGEN_ATTR_DATA_MARKER_EVENT_DATA_BIT_NUMBER**

data_marker_event_level_polarity

nifgen.Session.**data_marker_event_level_polarity**

Specifies the output polarity of the Data marker event.

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

Example: `my_session.data_markers[ ... ].data_marker_event_level_polarity`

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

Example: `my_session.data_marker_event_level_polarity`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.DataMarkerEventLevelPolarity
Permissions	read-write
Repeated Capabilities	data_markers

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

- LabVIEW Property: **Events:Data Marker:Level:Active Level**
- C Attribute: **NIFGEN_ATTR_DATA_MARKER_EVENT_LEVEL_POLARITY**

---

data_marker_event_output_terminal

nifgen.Session.**data_marker_event_output_terminal**

Specifies the destination terminal for the Data Marker Event.

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

Example: `my_session.data_markers[ ... ].data_marker_event_output_terminal`

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

Example: `my_session.data_marker_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	data_markers

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

- LabVIEW Property: **Events:Data Marker:Output Terminal**
- C Attribute: **NIFGEN_ATTR_DATA_MARKER_EVENT_OUTPUT_TERMINAL**

data_transfer_block_size

`nifgen.Session.`**`data_transfer_block_size`**

The number of samples at a time to download to onboard memory. Useful when the total data to be transferred to onboard memory is large.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Data Transfer Block Size**
- C Attribute: **NIFGEN_ATTR_DATA_TRANSFER_BLOCK_SIZE**

data_transfer_maximum_bandwidth

`nifgen.Session.`**`data_transfer_maximum_bandwidth`**

Specifies the maximum amount of bus bandwidth (in bytes per second) to use for data transfers. The signal generator limits data transfer speeds on the PCIe bus to the value you specify for this property. Set this property to optimize bus bandwidth usage for multi-device streaming applications by preventing the signal generator from consuming all of the available bandwidth on a PCI express link when waveforms are being written to the onboard memory of the device.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Maximum Bandwidth**
- C Attribute: **NIFGEN_ATTR_DATA_TRANSFER_MAXIMUM_BANDWIDTH**

data_transfer_maximum_in_flight_reads

nifgen.Session.**data_transfer_maximum_in_flight_reads**

Specifies the maximum number of concurrent PCI Express read requests the signal generator can issue. When transferring data from computer memory to device onboard memory across the PCI Express bus, the signal generator can issue multiple memory reads at the same time. In general, the larger the number of read requests, the more efficiently the device uses the bus because the multiple read requests keep the data flowing, even in a PCI Express topology that has high latency due to PCI Express switches in the data path. Most NI devices can issue a large number of read requests (typically 8 or 16). By default, this property is set to the highest value the signal generator supports. If other devices in your system cannot tolerate long data latencies, it may be helpful to decrease the number of in-flight read requests the NI signal generator issues. This helps to reduce the amount of data the signal generator reads at one time.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Advanced:Maximum In-Flight Read Requests**
- C Attribute: **NIFGEN_ATTR_DATA_TRANSFER_MAXIMUM_IN_FLIGHT_READS**

data_transfer_preferred_packet_size

nifgen.Session.**data_transfer_preferred_packet_size**

Specifies the preferred size of the data field in a PCI Express read request packet. In general, the larger the packet size, the more efficiently the device uses the bus. By default, NI signal generators use the largest packet size allowed by the system. However, due to different system implementations, some systems may perform better with smaller packet sizes. Recommended values for this property are powers of two between 64 and 512. In some cases, the signal generator generates packets smaller than the preferred size you set with this property. You cannot change this property while the device is generating a waveform. If you want to change the device configuration, call the *nifgen.Session.abort()* method or wait for the generation to complete.

Note: :

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Advanced:Preferred Packet Size**
- C Attribute: **NIFGEN_ATTR_DATA_TRANSFER_PREFERRED_PACKET_SIZE**

digital_data_mask

nifgen.Session.**digital_data_mask**

Specifies the mask to apply to the output on the digital connector. The masked data is replaced with the data in *nifgen.Session.digital_static_value*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Data Mask:Digital Data Mask**
- C Attribute: **NIFGEN_ATTR_DIGITAL_DATA_MASK**

digital_edge_script_trigger_edge

nifgen.Session.**digital_edge_script_trigger_edge**

Specifies the active edge for the Script trigger. This property is used when *nifgen.Session.script_trigger_type* is set to Digital Edge.

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

Example: `my_session.script_triggers[ ... ].digital_edge_script_trigger_edge`

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

Example: `my_session.digital_edge_script_trigger_edge`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ScriptTriggerDigitalEdgeEdge
Permissions	read-write
Repeated Capabilities	script_triggers

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

- LabVIEW Property: **Triggers:Script:Digital Edge:Edge**
- C Attribute: **NIFGEN_ATTR_DIGITAL_EDGE_SCRIPT_TRIGGER_EDGE**

digital_edge_script_trigger_source

`nifgen.Session.`**`digital_edge_script_trigger_source`**

Specifies the source terminal for the Script trigger. This property is used when *nifgen.Session.script_trigger_type* is set to Digital Edge.

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

Example: `my_session.script_triggers[ ... ].digital_edge_script_trigger_source`

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

Example: `my_session.digital_edge_script_trigger_source`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	script_triggers

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

- LabVIEW Property: **Triggers:Script:Digital Edge:Source**
- C Attribute: **NIFGEN_ATTR_DIGITAL_EDGE_SCRIPT_TRIGGER_SOURCE**

digital_edge_start_trigger_edge

`nifgen.Session.`**`digital_edge_start_trigger_edge`**

Specifies the active edge for the Start trigger. This property is used only when *nifgen.Session.start_trigger_type* is set to Digital Edge.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.StartTriggerDigitalEdgeEdge
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Triggers:Start:Digital Edge:Edge**
- C Attribute: **NIFGEN_ATTR_DIGITAL_EDGE_START_TRIGGER_EDGE**

digital_edge_start_trigger_source

nifgen.Session.**digital_edge_start_trigger_source**

Specifies the source terminal for the Start trigger. This property is used only when *nifgen.Session.start_trigger_type* is set to Digital Edge.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Triggers:Start:Digital Edge:Source**
- C Attribute: **NIFGEN_ATTR_DIGITAL_EDGE_START_TRIGGER_SOURCE**

digital_filter_enabled

nifgen.Session.**digital_filter_enabled**

Controls whether the signal generator applies a digital filter to the output signal. This property is valid in arbitrary waveform, arbitrary sequence, and script modes. This property can also be used in standard method and frequency list modes for user-defined waveforms.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Filters:Digital Filter Enabled**
- C Attribute: **NIFGEN_ATTR_DIGITAL_FILTER_ENABLED**

digital_filter_interpolation_factor

nifgen.Session.**digital_filter_interpolation_factor**

This property only affects the device when *nifgen.Session.digital_filter_enabled* is set to True. If you do not set this property directly, NI-FGEN automatically selects the maximum interpolation factor allowed for the current sample rate. Valid values are 2, 4, and 8.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Filters:Digital Filter Interpolation Factor**
- C Attribute: **NIFGEN_ATTR_DIGITAL_FILTER_INTERPOLATION_FACTOR**

digital_gain

nifgen.Session.**digital_gain**

Specifies a factor by which the signal generator digitally multiplies generated data before converting it to an analog signal in the DAC. For a digital gain greater than 1.0, the product of digital gain times the generated data must be inside the range plus or minus 1.0 (assuming floating point data). If the product exceeds these limits, the signal generator clips the output signal, and an error results. Some signal generators support both digital gain and an analog gain (analog gain is specified with the *nifgen.Session.func_amplitude* property or the *nifgen.Session.arb_gain* property). Digital gain can be changed during generation without the glitches that may occur when changing analog gains, due to relay switching. However, the DAC output resolution is a method of analog gain, so only analog gain makes full use of the resolution of the DAC.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Digital Gain**
- C Attribute: **NIFGEN_ATTR_DIGITAL_GAIN**

digital_pattern_enabled

nifgen.Session.**digital_pattern_enabled**

Controls whether the signal generator generates a digital pattern of the output signal.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Advanced:Digital Pattern Enabled**
- C Attribute: **NIFGEN_ATTR_DIGITAL_PATTERN_ENABLED**

digital_static_value

nifgen.Session.**digital_static_value**

Specifies the static value that replaces data masked by *nifgen.Session.digital_data_mask*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Data Mask:Digital Static Value**
- C Attribute: **NIFGEN_ATTR_DIGITAL_STATIC_VALUE**

done_event_output_terminal

nifgen.Session.**done_event_output_terminal**

Specifies the destination terminal for the Done Event.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Events:Done:Output Terminal**
- C Attribute: **NIFGEN_ATTR_DONE_EVENT_OUTPUT_TERMINAL**

driver_setup

`nifgen.Session.`**`driver_setup`**

Specifies the driver setup portion of the option string that was passed into the `nifgen.Session.InitWithOptions()` method.

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

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- C Attribute: **NIFGEN_ATTR_DRIVER_SETUP**

exported_onboard_reference_clock_output_terminal

`nifgen.Session.`**`exported_onboard_reference_clock_output_terminal`**

Specifies the terminal to which to export the Onboard Reference Clock.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Reference Clock:Onboard Reference Clock:Export Output Terminal**
- C Attribute: **NIFGEN_ATTR_EXPORTED_ONBOARD_REFERENCE_CLOCK_OUTPUT_TERMINAL**

exported_reference_clock_output_terminal

nifgen.Session.**exported_reference_clock_output_terminal**

Specifies the terminal to which to export the Reference Clock.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Reference Clock:Export Output Terminal**
- C Attribute: **NIFGEN_ATTR_EXPORTED_REFERENCE_CLOCK_OUTPUT_TERMINAL**

exported_sample_clock_divisor

nifgen.Session.**exported_sample_clock_divisor**

Specifies the factor by which to divide the Sample clock, also known as the Update clock, before it is exported. To export the Sample clock, use the `nifgen.Session.ExportSignal()` method or the *nifgen.Session.exported_sample_clock_output_terminal* property.

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

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock:Exported Sample Clock Divisor**
- C Attribute: **NIFGEN_ATTR_EXPORTED_SAMPLE_CLOCK_DIVISOR**

exported_sample_clock_output_terminal

nifgen.Session.**exported_sample_clock_output_terminal**

Specifies the terminal to which to export the Sample Clock.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock:Export Output Terminal**
- C Attribute: **NIFGEN_ATTR_EXPORTED_SAMPLE_CLOCK_OUTPUT_TERMINAL**

exported_sample_clock_timebase_divisor

nifgen.Session.**exported_sample_clock_timebase_divisor**

Specifies the factor by which to divide the sample clock timebase (board clock) before it is exported. To export the Sample clock timebase, use the `nifgen.Session.ExportSignal()` method or the *nifgen.Session.exported_sample_clock_timebase_output_terminal* property.

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

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock Timebase:Exported Sample Clock Timebase Divisor**
- C Attribute: **NIFGEN_ATTR_EXPORTED_SAMPLE_CLOCK_TIMEBASE_DIVISOR**

exported_sample_clock_timebase_output_terminal

nifgen.Session.**exported_sample_clock_timebase_output_terminal**

Specifies the terminal to which to export the Sample clock timebase. If you specify a divisor with the *nifgen.Session.exported_sample_clock_timebase_divisor* property, the Sample clock exported with the *nifgen.Session.exported_sample_clock_timebase_output_terminal* property is the value of the Sample clock timebase after it is divided-down. For a list of the terminals available on your device, refer to the Device Routes tab in MAX. To change the device configuration, call *nifgen.Session.abort()* or wait for the generation to complete.

Note: The signal generator must not be in the Generating state when you change this property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock Timebase:Export Output Terminal**
- C Attribute: **NIFGEN_ATTR_EXPORTED_SAMPLE_CLOCK_TIMEBASE_OUTPUT_TERMINAL**

---

exported_script_trigger_output_terminal

nifgen.Session.**exported_script_trigger_output_terminal**

Specifies the output terminal for the exported Script trigger. Setting this property to an empty string means that when you commit the session, the signal is removed from that terminal and, if possible, the terminal is tristated.

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

Example: `my_session.script_triggers[ ... ].exported_script_trigger_output_terminal`

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

Example: `my_session.exported_script_trigger_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	script_triggers

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

- LabVIEW Property: **Triggers:Script:Output Terminal**
- C Attribute: **NIFGEN_ATTR_EXPORTED_SCRIPT_TRIGGER_OUTPUT_TERMINAL**

exported_start_trigger_output_terminal

`nifgen.Session.exported_start_trigger_output_terminal`

Specifies the destination terminal for exporting the Start trigger.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Triggers:Start:Output Terminal**
- C Attribute: **NIFGEN_ATTR_EXPORTED_START_TRIGGER_OUTPUT_TERMINAL**

external_clock_delay_binary_value

`nifgen.Session.external_clock_delay_binary_value`

Binary value of the external clock delay.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Advanced:External Clock Delay Binary Value**
- C Attribute: **NIFGEN_ATTR_EXTERNAL_CLOCK_DELAY_BINARY_VALUE**

external_sample_clock_multiplier

nifgen.Session.**external_sample_clock_multiplier**

Specifies a multiplication factor to use to obtain a desired sample rate from an external Sample clock. The resulting sample rate is equal to this factor multiplied by the external Sample clock rate. You can use this property to generate samples at a rate higher than your external clock rate. When using this property, you do not need to explicitly set the external clock rate.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Advanced:External Sample Clock Multiplier**
- C Attribute: **NIFGEN_ATTR_EXTERNAL_SAMPLE_CLOCK_MULTIPLIER**

file_transfer_block_size

nifgen.Session.**file_transfer_block_size**

The number of samples at a time to read from the file and download to onboard memory. Used in conjunction with the Create From File and Write From File methods.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:File Transfer Block Size**
- C Attribute: **NIFGEN_ATTR_FILE_TRANSFER_BLOCK_SIZE**

filter_correction_frequency

nifgen.Session.**filter_correction_frequency**

Controls the filter correction frequency of the analog filter. This property corrects for the ripples in the analog filter frequency response at the frequency specified. For standard waveform output, the filter correction frequency should be set to be the same as the frequency of the standard waveform. To have no filter correction, set this property to 0 Hz.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:5401/5411/5431:Filter Correction Frequency**
- C Attribute: **NIFGEN_ATTR_FILTER_CORRECTION_FREQUENCY**

flatness_correction_enabled

nifgen.Session.**flatness_correction_enabled**

When True, the signal generator applies a flatness correction factor to the generated sine wave in order to ensure the same output power level at all frequencies. This property should be set to False when performing Flatness Calibration.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Filters:Flatness Correction Enabled**
- C Attribute: **NIFGEN_ATTR_FLATNESS_CORRECTION_ENABLED**

fpga_bitfile_path

nifgen.Session.**fpga_bitfile_path**

Gets the absolute file path to the bitfile loaded on the FPGA.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:FPGA Bitfile Path**
- C Attribute: **NIFGEN_ATTR_FPGA_BITFILE_PATH**

freq_list_duration_quantum

nifgen.Session.**freq_list_duration_quantum**

Returns the quantum of which all durations must be a multiple in a frequency list.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Frequency List Mode:Frequency List Duration Quantum**
- C Attribute: **NIFGEN_ATTR_FREQ_LIST_DURATION_QUANTUM**

freq_list_handle

nifgen.Session.**freq_list_handle**

Sets which frequency list the signal generator produces. Create a frequency list using *nifgen.Session.create_freq_list()*. *nifgen.Session.create_freq_list()* returns a handle that you can use to identify the list.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Frequency List Mode:Frequency List Handle**
- C Attribute: **NIFGEN_ATTR_FREQ_LIST_HANDLE**

func_amplitude

`nifgen.Session.`**`func_amplitude`**

Controls the amplitude of the standard waveform that the signal generator produces. This value is the amplitude at the output terminal. For example, to produce a waveform ranging from -5.00 V to +5.00 V, set the amplitude to 10.00 V. set the Waveform parameter to *DC*. Units: Vpk-pk

Note: This parameter does not affect signal generator behavior when you

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

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

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

Example: `my_session.func_amplitude`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Standard Function:Amplitude**
- C Attribute: **NIFGEN_ATTR_FUNC_AMPLITUDE**

func_buffer_size

nifgen.Session.**func_buffer_size**

This property contains the number of samples used in the standard method waveform buffer. This property is only valid on devices that implement standard method mode in software, and is read-only for all other devices. implementation of Standard Method Mode on your device.

Note: Refer to the Standard Method Mode topic for more information on the

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Standard Function Mode:Buffer Size**
- C Attribute: **NIFGEN_ATTR_FUNC_BUFFER_SIZE**

func_dc_offset

nifgen.Session.**func_dc_offset**

Controls the DC offset of the standard waveform that the signal generator produces. This value is the offset at the output terminal. The value is the offset from ground to the center of the waveform that you specify with the Waveform parameter. For example, to configure a waveform with an amplitude of 10.00 V to range from 0.00 V to +10.00 V, set DC Offset to 5.00 V. Units: volts

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

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

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

Example: `my_session.func_dc_offset`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Standard Function:DC Offset**
- C Attribute: **NIFGEN_ATTR_FUNC_DC_OFFSET**

func_duty_cycle_high

nifgen.Session.**func_duty_cycle_high**

Controls the duty cycle of the square wave the signal generator produces. Specify this property as a percentage of the time the square wave is high in a cycle. set the Waveform parameter to *SQUARE*. Units: Percentage of time the waveform is high

Note: This parameter only affects signal generator behavior when you

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

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

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

Example: `my_session.func_duty_cycle_high`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Standard Function:Duty Cycle High**
- C Attribute: **NIFGEN_ATTR_FUNC_DUTY_CYCLE_HIGH**

func_frequency

nifgen.Session.**func_frequency**

Controls the frequency of the standard waveform that the signal generator produces. Units: hertz (1) This parameter does not affect signal generator behavior when you set the Waveform parameter of the *nifgen.Session.configure_standard_waveform()* method to *DC*. (2) For *SINE*, the range is between 0 MHz and 16 MHz, but the range is between 0 MHz and 1 MHz for all other waveforms.

Note: :

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

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

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

Example: `my_session.func_frequency`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Standard Function:Standard Function Mode:Frequency**
- C Attribute: **NIFGEN_ATTR_FUNC_FREQUENCY**

func_max_buffer_size

`nifgen.Session.`**`func_max_buffer_size`**

This property sets the maximum number of samples that can be used in the standard method waveform buffer. Increasing this value may increase the quality of the waveform. This property is only valid on devices that implement standard method mode in software, and is read-only for all other devices. implementation of Standard Method Mode on your device.

Note: Refer to the Standard Method Mode topic for more information on the

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Standard Function Mode:Maximum Buffer Size**
- C Attribute: **NIFGEN_ATTR_FUNC_MAX_BUFFER_SIZE**

func_start_phase

nifgen.Session.**func_start_phase**

Controls horizontal offset of the standard waveform the signal generator produces. Specify this property in degrees of one waveform cycle. A start phase of 180 degrees means output generation begins halfway through the waveform. A start phase of 360 degrees offsets the output by an entire waveform cycle, which is identical to a start phase of 0 degrees. set the Waveform parameter to *DC*. Units: Degrees of one cycle

---

Note: This parameter does not affect signal generator behavior when you

---

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

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

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

Example: `my_session.func_start_phase`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Standard Function:Start Phase**
- C Attribute: **NIFGEN_ATTR_FUNC_START_PHASE**

---

func_waveform

nifgen.Session.**func_waveform**

This channel-based property specifies which standard waveform the signal generator produces. Use this property only when *nifgen.Session.output_mode* is set to *FUNC*. *SINE* - Sinusoid waveform *SQUARE* - Square waveform *TRIANGLE* - Triangle waveform *RAMP_UP* - Positive ramp waveform *RAMP_DOWN* - Negative ramp waveform *DC* - Constant voltage *NOISE* - White noise *USER* - User-defined waveform as defined with *nifgen.Session.define_user_standard_waveform()*

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

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

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

Example: `my_session.func_waveform`

The following table lists the characteristics of this property.

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

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

- LabVIEW Property: **Standard Function:Waveform**
- C Attribute: **NIFGEN_ATTR_FUNC_WAVEFORM**

idle_behavior

`nifgen.Session.idle_behavior`

Specifies the behavior of the output during the Idle state. The output can be configured to hold the last generated voltage before entering the Idle state or jump to the Idle Value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.IdleBehavior
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Advanced:Idle Behavior**
- C Attribute: **NIFGEN_ATTR_IDLE_BEHAVIOR**

idle_value

`nifgen.Session.idle_value`

Specifies the value to generate in the Idle state. The Idle Behavior must be configured to jump to this value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Advanced:Idle Value**
- C Attribute: **NIFGEN_ATTR_IDLE_VALUE**

instrument_firmware_revision

nifgen.Session.**instrument_firmware_revision**

A string that contains the firmware revision information for the device that you are currently using.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

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

instrument_manufacturer

nifgen.Session.**instrument_manufacturer**

A string that contains the name of the device manufacturer you are currently using.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

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

instrument_model

nifgen.Session.**instrument_model**

A string that contains the model number or name of the device that you are currently using.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

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

io_resource_descriptor

nifgen.Session.**io_resource_descriptor**

Indicates the resource descriptor that NI-FGEN uses to identify the physical device. If you initialize NI-FGEN with a logical name, this property contains the resource descriptor that corresponds to the entry in the IVI Configuration Utility. If you initialize NI-FGEN with the resource descriptor, this property contains that value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Inherent IVI Attributes:Advanced Session Information:Resource Descriptor**
- C Attribute: **NIFGEN_ATTR_IO_RESOURCE_DESCRIPTOR**

load_impedance

nifgen.Session.**load_impedance**

This channel-based property specifies the load impedance connected to the analog output of the channel. If you set this property to NIFGEN_VAL_MATCHED_LOAD_IMPEDANCE (-1.0), NI-FGEN assumes that the load impedance matches the output impedance. NI-FGEN compensates to give the desired peak-to-peak voltage amplitude or arbitrary gain (relative to 1 V).

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
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Load Impedance**
- C Attribute: **NIFGEN_ATTR_LOAD_IMPEDANCE**

---

logical_name

nifgen.Session.**logical_name**

A string containing the logical name that you specified when opening the current IVI session. You may pass a logical name to nifgen.Session.init() or nifgen.Session.InitWithOptions(). The IVI Configuration Utility must contain an entry for the logical name. The logical name entry refers to a virtual instrument section in the IVI Configuration file. The virtual instrument section specifies a physical device and initial user options.

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

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Inherent IVI Attributes:Advanced Session Information:Logical Name**

- C Attribute: **NIFGEN_ATTR_LOGICAL_NAME**

marker_events_count

nifgen.Session.**marker_events_count**

Returns the number of markers supported by the device. Use this property when *nifgen.Session.output_mode* is set to *SCRIPT*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Marker Events Count**
- C Attribute: **NIFGEN_ATTR_MARKER_EVENTS_COUNT**

marker_event_output_terminal

nifgen.Session.**marker_event_output_terminal**

Specifies the destination terminal for the Marker Event.

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

Example: `my_session.markers[ ... ].marker_event_output_terminal`

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

Example: `my_session.marker_event_output_terminal`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	markers

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

- LabVIEW Property: **Events:Marker:Output Terminal**
- C Attribute: **NIFGEN_ATTR_MARKER_EVENT_OUTPUT_TERMINAL**

max_freq_list_duration

nifgen.Session.**max_freq_list_duration**

Returns the maximum duration of any one step in the frequency list.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Frequency List Mode:Maximum Frequency List Duration**
- C Attribute: **NIFGEN_ATTR_MAX_FREQ_LIST_DURATION**

max_freq_list_length

nifgen.Session.**max_freq_list_length**

Returns the maximum number of steps that can be in a frequency list.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Frequency List Mode:Maximum Frequency List Length**
- C Attribute: **NIFGEN_ATTR_MAX_FREQ_LIST_LENGTH**

max_loop_count

nifgen.Session.**max_loop_count**

Returns the maximum number of times that the signal generator can repeat a waveform in a sequence. Typically, this value is constant for the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Sequence Mode:Max Loop Count**
- C Attribute: **NIFGEN_ATTR_MAX_LOOP_COUNT**

max_num_freq_lists

`nifgen.Session.max_num_freq_lists`

Returns the maximum number of frequency lists the signal generator allows.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Frequency List Mode:Maximum Number Of Frequency Lists**
- C Attribute: **NIFGEN_ATTR_MAX_NUM_FREQ_LISTS**

max_num_sequences

`nifgen.Session.max_num_sequences`

Returns the maximum number of arbitrary sequences that the signal generator allows. Typically, this value is constant for the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Sequence Mode:Max Number of Sequences**
- C Attribute: **NIFGEN_ATTR_MAX_NUM_SEQUENCES**

max_num_waveforms

nifgen.Session.**max_num_waveforms**

Returns the maximum number of arbitrary waveforms that the signal generator allows. Typically, this value is constant for the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Capabilities:Max Number of Waveforms**
- C Attribute: **NIFGEN_ATTR_MAX_NUM_WAVEFORMS**

max_sequence_length

nifgen.Session.**max_sequence_length**

Returns the maximum number of arbitrary waveforms that the signal generator allows in a sequence. Typically, this value is constant for the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Sequence Mode:Max Sequence Length**
- C Attribute: **NIFGEN_ATTR_MAX_SEQUENCE_LENGTH**

max_waveform_size

nifgen.Session.**max_waveform_size**

Returns the size, in samples, of the largest waveform that can be created. This property reflects the space currently available, taking into account previously allocated waveforms and instructions.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Capabilities:Max Waveform Size**
- C Attribute: **NIFGEN_ATTR_MAX_WAVEFORM_SIZE**

memory_size

nifgen.Session.**memory_size**

The total amount of memory, in bytes, on the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Memory Size**
- C Attribute: **NIFGEN_ATTR_MEMORY_SIZE**

min_freq_list_duration

nifgen.Session.**min_freq_list_duration**

Returns the minimum number of steps that can be in a frequency list.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Frequency List Mode:Minimum Frequency List Duration**
- C Attribute: **NIFGEN_ATTR_MIN_FREQ_LIST_DURATION**

min_freq_list_length

nifgen.Session.**min_freq_list_length**

Returns the minimum number of frequency lists that the signal generator allows.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Standard Function:Frequency List Mode:Minimum Frequency List Length**
- C Attribute: **NIFGEN_ATTR_MIN_FREQ_LIST_LENGTH**

min_sequence_length

nifgen.Session.**min_sequence_length**

Returns the minimum number of arbitrary waveforms that the signal generator allows in a sequence. Typically, this value is constant for the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Arbitrary Sequence Mode:Min Sequence Length**
- C Attribute: **NIFGEN_ATTR_MIN_SEQUENCE_LENGTH**

min_waveform_size

nifgen.Session.**min_waveform_size**

Returns the minimum number of points that the signal generator allows in an arbitrary waveform. Typically, this value is constant for the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Capabilities:Min Waveform Size**
- C Attribute: **NIFGEN_ATTR_MIN_WAVEFORM_SIZE**

module_revision

nifgen.Session.**module_revision**

A string that contains the module revision for the device that you are currently using.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Inherent IVI Attributes:Instrument Identification:Module Revision**
- C Attribute: **NIFGEN_ATTR_MODULE_REVISION**

channel_count

nifgen.Session.**channel_count**

Indicates the number of channels that the specific instrument driver supports. For each property for which IVI_VAL_MULTI_CHANNEL 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
Repeated Capabilities	None

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

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

output_enabled

nifgen.Session.**output_enabled**

This channel-based property specifies whether the signal that the signal generator produces appears at the output connector.

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

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

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

Example: `my_session.output_enabled`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Output:Output Enabled**
- C Attribute: **NIFGEN_ATTR_OUTPUT_ENABLED**

output_impedance

nifgen.Session.**output_impedance**

This channel-based property specifies the signal generator output impedance at the output connector. NI signal sources modules have an output impedance of 50 ohms and an optional 75 ohms on select modules. If the load impedance matches the output impedance, then the voltage at the signal output connector is at the needed level. The voltage at the signal output connector varies with load output impedance, up to doubling the voltage for a high-impedance load.

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

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

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

Example: `my_session.output_impedance`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	channels

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

- LabVIEW Property: **Output:Output Impedance**
- C Attribute: **NIFGEN_ATTR_OUTPUT_IMPEDANCE**

---

output_mode

nifgen.Session.**output_mode**

Sets which output mode the signal generator will use. The value you specify determines which methods and properties you use to configure the waveform the signal generator produces.

Note: The signal generator must not be in the Generating state when you change this property. To change the device configuration, call *nifgen.Session.abort()* or wait for the generation to complete.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.OutputMode
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Output Mode**
- C Attribute: **NIFGEN_ATTR_OUTPUT_MODE**

ready_for_start_event_output_terminal

nifgen.Session.**ready_for_start_event_output_terminal**

Specifies the destination terminal for the Ready for Start Event.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Events:Ready For Start:Output Terminal**
- C Attribute: **NIFGEN_ATTR_READY_FOR_START_EVENT_OUTPUT_TERMINAL**

reference_clock_source

nifgen.Session.**reference_clock_source**

Specifies the reference clock source used by the signal generator. The signal generator derives the frequencies and sample rates that it uses to generate waveforms from the source you specify. For example, when you set this property to ClkIn, the signal generator uses the signal it receives at the CLK IN front panel connector as the Reference clock. To change the device configuration, call *nifgen.Session.abort()* or wait for the generation to complete.

Note: The signal generator must not be in the Generating state when you change this property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ReferenceClockSource
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Reference Clock:Source**

- C Attribute: **NIFGEN_ATTR_REFERENCE_CLOCK_SOURCE**

ref_clock_frequency

nifgen.Session.**ref_clock_frequency**

Sets the frequency of the signal generator reference clock. The signal generator uses the reference clock to derive frequencies and sample rates when generating output.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Reference Clock:Frequency**
- C Attribute: **NIFGEN_ATTR_REF_CLOCK_FREQUENCY**

sample_clock_source

nifgen.Session.**sample_clock_source**

Specifies the Sample clock source. If you specify a divisor with the *nifgen.Session.exported_sample_clock_divisor* property, the Sample clock exported with the *nifgen.Session.exported_sample_clock_output_terminal* property is the value of the Sample clock after it is divided-down. For a list of the terminals available on your device, refer to the Device Routes tab in MAX. To change the device configuration, call *nifgen.Session.abort()* or wait for the generation to complete.

Note: The signal generator must not be in the Generating state when you change this property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.SampleClockSource
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock:Source**
- C Attribute: **NIFGEN_ATTR_SAMPLE_CLOCK_SOURCE**

sample_clock_timebase_rate

nifgen.Session.**sample_clock_timebase_rate**

Specifies the Sample clock timebase rate. This property applies only to external Sample clock timebases. To change the device configuration, call *nifgen.Session.abort()* or wait for the generation to complete.

Note: The signal generator must not be in the Generating state when you change this property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock Timebase:Rate**
- C Attribute: **NIFGEN_ATTR_SAMPLE_CLOCK_TIMEBASE_RATE**

sample_clock_timebase_source

nifgen.Session.**sample_clock_timebase_source**

Specifies the Sample Clock Timebase source. To change the device configuration, call the *nifgen.Session.abort()* method or wait for the generation to complete.

Note: The signal generator must not be in the Generating state when you change this property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.SampleClockTimebaseSource
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Clocks:Sample Clock Timebase:Source**
- C Attribute: **NIFGEN_ATTR_SAMPLE_CLOCK_TIMEBASE_SOURCE**

script_to_generate

nifgen.Session.**script_to_generate**

Specifies which script the generator produces. To configure the generator to run a particular script, set this property to the name of the script. Use *nifgen.Session.write_script()* to create multiple scripts. Use this property when *nifgen.Session.output_mode* is set to *SCRIPT*.

Note: The signal generator must not be in the Generating state when you change this property. To change the device configuration, call *nifgen.Session.abort()* or wait for the generation to complete.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Script Mode:Script to Generate**
- C Attribute: **NIFGEN_ATTR_SCRIPT_TO_GENERATE**

script_triggers_count

nifgen.Session.**script_triggers_count**

Specifies the number of Script triggers supported by the device. Use this property when *nifgen.Session.output_mode* is set to *SCRIPT*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Script Triggers Count**
- C Attribute: **NIFGEN_ATTR_SCRIPT_TRIGGERS_COUNT**

script_trigger_type

nifgen.Session.**script_trigger_type**

Specifies the Script trigger type. Depending upon the value of this property, additional properties may need to be configured to fully configure the trigger.

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

Example: `my_session.script_triggers[ ... ].script_trigger_type`

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

Example: `my_session.script_trigger_type`

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.ScriptTriggerType
Permissions	read-write
Repeated Capabilities	script_triggers

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

- LabVIEW Property: **Triggers:Script:Trigger Type**
- C Attribute: **NIFGEN_ATTR_SCRIPT_TRIGGER_TYPE**

---

serial_number

nifgen.Session.**serial_number**

The signal generator's serial number.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Serial Number**
- C Attribute: **NIFGEN_ATTR_SERIAL_NUMBER**

---

simulate

nifgen.Session.**simulate**

Specifies whether to simulate NI-FGEN I/O operations. If simulation is enabled, NI-FGEN methods perform range checking and call Ivi_GetAttribute and Ivi_SetAttribute, but they do not perform device I/O. For output parameters that represent device data, NI-FGEN methods return calculated values. Default Value: False Use `nifgen.Session.InitWithOptions()` to override default value.

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

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	bool
Permissions	read-write
Repeated Capabilities	None

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

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

specific_driver_description

nifgen.Session.**specific_driver_description**

Returns a brief description of NI-FGEN.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Inherent IVI Attributes:Driver Identification:Description**
- C Attribute: **NIFGEN_ATTR_SPECIFIC_DRIVER_DESCRIPTION**

major_version

nifgen.Session.**major_version**

Returns the major version number of NI-FGEN.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Obsolete:Major Version**
- C Attribute: **NIFGEN_ATTR_SPECIFIC_DRIVER_MAJOR_VERSION**

minor_version

nifgen.Session.**minor_version**

Returns the minor version number of NI-FGEN.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Obsolete:Minor Version**
- C Attribute: **NIFGEN_ATTR_SPECIFIC_DRIVER_MINOR_VERSION**

specific_driver_revision

nifgen.Session.**specific_driver_revision**

A string that contains additional version information about NI-FGEN.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Inherent IVI Attributes:Driver Identification:Revision**
- C Attribute: **NIFGEN_ATTR_SPECIFIC_DRIVER_REVISION**

specific_driver_vendor

nifgen.Session.**specific_driver_vendor**

A string that contains the name of the vendor that supplies NI-FGEN.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Instrument:Inherent IVI Attributes:Driver Identification:Driver Vendor**
- C Attribute: **NIFGEN_ATTR_SPECIFIC_DRIVER_VENDOR**

started_event_output_terminal

nifgen.Session.**started_event_output_terminal**

Specifies the destination terminal for the Started Event.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Events:Started:Output Terminal**
- C Attribute: **NIFGEN_ATTR_STARTED_EVENT_OUTPUT_TERMINAL**

start_trigger_type

nifgen.Session.**start_trigger_type**

Specifies whether you want the Start trigger to be a Digital Edge, or Software trigger. You can also choose None as the value for this property.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.StartTriggerType
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Triggers:Start:Trigger Type**
- C Attribute: **NIFGEN_ATTR_START_TRIGGER_TYPE**

streaming_space_available_in_waveform

nifgen.Session.**streaming_space_available_in_waveform**

Indicates the space available (in samples) in the streaming waveform for writing new data. During generation, this available space may be in multiple locations with, for example, part of the available space at the end of the streaming waveform and the rest at the beginning. In this situation, writing a block of waveform data the size of the total space available in the streaming waveform causes NI-FGEN to return an error, as NI-FGEN will not wrap the data from the end of the waveform to the beginning and cannot write data past the end of the waveform buffer. To avoid writing data past the end of the waveform, write new data to the waveform in a fixed size that is an integer divisor of the total size of the streaming waveform. Used in conjunction with the *nifgen.Session.streaming_waveform_handle* or *nifgen.Session.streaming_waveform_name* properties.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Streaming:Space Available in Streaming Waveform**
- C Attribute: **NIFGEN_ATTR_STREAMING_SPACE_AVAILABLE_IN_WAVEFORM**

streaming_waveform_handle

nifgen.Session.**streaming_waveform_handle**

Specifies the waveform handle of the waveform used to continuously stream data during generation. This property defaults to -1 when no streaming waveform is specified. Used in conjunction with *nifgen.Session.streaming_space_available_in_waveform*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Streaming:Streaming Waveform Handle**
- C Attribute: **NIFGEN_ATTR_STREAMING_WAVEFORM_HANDLE**

streaming_waveform_name

nifgen.Session.**streaming_waveform_name**

Specifies the name of the waveform used to continuously stream data during generation. This property defaults to // when no streaming waveform is specified. Use in conjunction with *nifgen.Session.streaming_space_available_in_waveform*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Streaming:Streaming Waveform Name**
- C Attribute: **NIFGEN_ATTR_STREAMING_WAVEFORM_NAME**

streaming_write_timeout

nifgen.Session.**streaming_write_timeout**

Specifies the maximum amount of time allowed to complete a streaming write operation.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Data Transfer:Streaming:Streaming Write Timeout**
- C Attribute: **NIFGEN_ATTR_STREAMING_WRITE_TIMEOUT**

supported_instrument_models

nifgen.Session.**supported_instrument_models**

Returns a model code of the device. For NI-FGEN versions that support more than one device, this property contains a comma-separated list of supported device models.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read only
Repeated Capabilities	None

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

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

terminal_configuration

nifgen.Session.**terminal_configuration**

Specifies whether gain and offset values will be analyzed based on single-ended or differential operation.

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

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

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

Example: `my_session.terminal_configuration`

The following table lists the characteristics of this property.

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

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

- LabVIEW Property: **Output:Terminal Configuration**
- C Attribute: **NIFGEN_ATTR_TERMINAL_CONFIGURATION**

---

trigger_mode

nifgen.Session.**trigger_mode**

Controls the trigger mode.

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

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

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

Example: `my_session.trigger_mode`

The following table lists the characteristics of this property.

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

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

- LabVIEW Property: **Triggers:Trigger Mode**
- C Attribute: **NIFGEN_ATTR_TRIGGER_MODE**

wait_behavior

`nifgen.Session.wait_behavior`

Specifies the behavior of the output while waiting for a script trigger or during a wait instruction. The output can be configured to hold the last generated voltage before waiting or jump to the Wait Value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	enums.WaitBehavior
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Advanced:Wait Behavior**
- C Attribute: **NIFGEN_ATTR_WAIT_BEHAVIOR**

wait_value

`nifgen.Session.wait_value`

Specifies the value to generate while waiting. The Wait Behavior must be configured to jump to this value.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read-write
Repeated Capabilities	None

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

- LabVIEW Property: **Output:Advanced:Wait Value**
- C Attribute: **NIFGEN_ATTR_WAIT_VALUE**

waveform_quantum

nifgen.Session.**waveform_quantum**

The size of each arbitrary waveform must be a multiple of a quantum value. This property returns the quantum value that the signal generator allows. For example, when this property returns a value of 8, all waveform sizes must be a multiple of 8. Typically, this value is constant for the signal generator.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	int
Permissions	read only
Repeated Capabilities	None

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

- LabVIEW Property: **Arbitrary Waveform:Capabilities:Waveform Quantum**
- C Attribute: **NIFGEN_ATTR_WAVEFORM_QUANTUM**

NI-TClk Support

nifgen.Session.**tclk**

This is used to get and set NI-TClk attributes on the session.

See also:

See `nitclk.SessionReference` for a complete list of attributes.

Session

- *configure_freq_list*
- *configure_standard_waveform*
- *create_advanced_arb_sequence*
- *create_arb_sequence*
- *create_freq_list*
- *create_waveform_from_file_f64*
- *create_waveform_from_file_i16*
- *create_waveform_numpy*
- *define_user_standard_waveform*
- *delete_script*
- *delete_waveform*
- *disable*
- *export_attribute_configuration_buffer*
- *export_attribute_configuration_file*
- *get_channel_name*
- *get_ext_cal_last_date_and_time*
- *get_ext_cal_last_temp*
- *get_ext_cal_recommended_interval*
- *get_hardware_state*
- *get_self_cal_last_date_and_time*
- *get_self_cal_last_temp*
- *get_self_cal_supported*
- *import_attribute_configuration_buffer*
- *import_attribute_configuration_file*
- *initiate*
- *is_done*
- *lock*
- *query_arb_seq_capabilities*
- *query_arb_wfm_capabilities*
- *query_freq_list_capabilities*
- *read_current_temperature*
- *reset*
- *reset_device*
- *reset_with_defaults*
- *self_cal*

- *digital_data_mask*
- *digital_edge_script_trigger_edge*
- *digital_edge_script_trigger_source*
- *digital_edge_start_trigger_edge*
- *digital_edge_start_trigger_source*
- *digital_filter_enabled*
- *digital_filter_interpolation_factor*
- *digital_gain*
- *digital_pattern_enabled*
- *digital_static_value*
- *done_event_output_terminal*
- *driver_setup*
- *exported_onboard_reference_clock_output_terminal*
- *exported_reference_clock_output_terminal*
- *exported_sample_clock_divisor*
- *exported_sample_clock_output_terminal*
- *exported_sample_clock_timebase_divisor*
- *exported_sample_clock_timebase_output_terminal*
- *exported_script_trigger_output_terminal*
- *exported_start_trigger_output_terminal*
- *external_clock_delay_binary_value*
- *external_sample_clock_multiplier*
- *file_transfer_block_size*
- *filter_correction_frequency*
- *flatness_correction_enabled*
- *fpga_bitfile_path*
- *freq_list_duration_quantum*
- *freq_list_handle*
- *func_amplitude*
- *func_buffer_size*
- *func_dc_offset*
- *func_duty_cycle_high*
- *func_frequency*
- *func_max_buffer_size*
- *func_start_phase*

- *func_waveform*
- *idle_behavior*
- *idle_value*
- *instrument_firmware_revision*
- *instrument_manufacturer*
- *instrument_model*
- *io_resource_descriptor*
- *load_impedance*
- *logical_name*
- *marker_events_count*
- *marker_event_output_terminal*
- *max_freq_list_duration*
- *max_freq_list_length*
- *max_loop_count*
- *max_num_freq_lists*
- *max_num_sequences*
- *max_num_waveforms*
- *max_sequence_length*
- *max_waveform_size*
- *memory_size*
- *min_freq_list_duration*
- *min_freq_list_length*
- *min_sequence_length*
- *min_waveform_size*
- *module_revision*
- *channel_count*
- *output_enabled*
- *output_impedance*
- *output_mode*
- *ready_for_start_event_output_terminal*
- *reference_clock_source*
- *ref_clock_frequency*
- *sample_clock_source*
- *sample_clock_timebase_rate*
- *sample_clock_timebase_source*

Repeated Capabilities

Repeated capabilities attributes are used to set the *channel_string* parameter to the underlying driver function call. This can be the actual function based on the `Session` method being called, or it can be the appropriate Get/Set Attribute function, such as `niFgen_SetAttributeViInt32()`.

Repeated capabilities attributes use the indexing operator `[]` to indicate the repeated capabilities. The parameter can be a string, list, tuple, or slice (range). Each element of those can be a string or an integer. If it is a string, you can indicate a range using the same format as the driver: `'0-2'` or `'0:2'`

Some repeated capabilities use a prefix before the number and this is optional

channels

nifgen.Session.**channels**

```
session.channels['0-2'].channel_enabled = True
```

passes a string of `'0, 1, 2'` to the set attribute function.

script_triggers

nifgen.Session.**script_triggers**

If no prefix is added to the items in the parameter, the correct prefix will be added when the driver function call is made.

```
session.script_triggers['0-2'].channel_enabled = True
```

passes a string of `'ScriptTrigger0, ScriptTrigger1, ScriptTrigger2'` to the set attribute function.

If an invalid repeated capability is passed to the driver, the driver will return an error.

You can also explicitly use the prefix as part of the parameter, but it must be the correct prefix for the specific repeated capability.

```
session.script_triggers['ScriptTrigger0-ScriptTrigger2'].channel_enabled =␣
↪True
```

passes a string of `'ScriptTrigger0, ScriptTrigger1, ScriptTrigger2'` to the set attribute function.

markers

nifgen.Session.**markers**

If no prefix is added to the items in the parameter, the correct prefix will be added when the driver function call is made.

```
session.markers['0-2'].channel_enabled = True
```

passes a string of `'Marker0, Marker1, Marker2'` to the set attribute function.

If an invalid repeated capability is passed to the driver, the driver will return an error.

You can also explicitly use the prefix as part of the parameter, but it must be the correct prefix for the specific repeated capability.

```
session.markers['Marker0-Marker2'].channel_enabled = True
```

passes a string of `'Marker0, Marker1, Marker2'` to the set attribute function.

data_markers

nifgen.Session.**data_markers**

If no prefix is added to the items in the parameter, the correct prefix will be added when the driver function call is made.

```
session.data_markers['0-2'].channel_enabled = True
```

passes a string of `'DataMarker0, DataMarker1, DataMarker2'` to the set attribute function.

If an invalid repeated capability is passed to the driver, the driver will return an error.

You can also explicitly use the prefix as part of the parameter, but it must be the correct prefix for the specific repeated capability.

```
session.data_markers['DataMarker0-DataMarker2'].channel_enabled = True
```

passes a string of `'DataMarker0, DataMarker1, DataMarker2'` to the set attribute function.

Enums

Enums used in NI-FGEN

AnalogPath

class nifgen.**AnalogPath**

MAIN

Specifies use of the main path. NI-FGEN chooses the amplifier based on the user-specified gain.

DIRECT

Specifies use of the direct path.

FIXED_LOW_GAIN

Specifies use of the low-gain amplifier in the main path, no matter what value the user specifies for gain. This setting limits the output range.

FIXED_HIGH_GAIN

Specifies use of the high-gain amplifier in the main path.

BusType

class nifgen.**BusType**

INVALID

Indicates an invalid bus type.

AT

Indicates the signal generator is the AT bus type.

PCI

Indicates the signal generator is the PCI bus type.

PXI

Indicates the signal generator is the PXI bus type.

VXI

Indicates the signal generator is the VXI bus type.

PCMCIA

Indicates the signal generator is the PCI-CMA bus type.

PXIE

Indicates the signal generator is the PXI Express bus type.

ByteOrder

class nifgen.**ByteOrder**

LITTLE

BIG

ClockMode

class nifgen.**ClockMode**

HIGH_RESOLUTION

High resolution sampling—Sample rate is generated by a high–resolution clock source.

DIVIDE_DOWN

Divide down sampling—Sample rates are generated by dividing the source frequency.

AUTOMATIC

Automatic Selection—NI-FGEN selects between the divide–down and high–resolution clocking modes.

DataMarkerEventLevelPolarity

class nifgen.**DataMarkerEventLevelPolarity**

HIGH

When the operation is ready to start, the Ready for Start event level is high.

LOW

When the operation is ready to start, the Ready for Start event level is low.

HardwareState

class nifgen.**HardwareState**

IDLE

WAITING_FOR_START_TRIGGER

RUNNING

DONE

HARDWARE_ERROR

IdleBehavior

class nifgen.**IdleBehavior**

HOLD_LAST

While in an Idle or Wait state, the output signal remains at the last voltage generated prior to entering the state.

JUMP_TO

While in an Idle or Wait state, the output signal remains at the value configured in the Idle or Wait value property.

OutputMode

class nifgen.**OutputMode**

FUNC

Standard Method mode— Generates standard method waveforms such as sine, square, triangle, and so on.

ARB

Arbitrary waveform mode—Generates waveforms from user-created/provided waveform arrays of numeric data.

SEQ

Arbitrary sequence mode — Generates downloaded waveforms in an order your specify.

FREQ_LIST

Frequency List mode—Generates a standard method using a list of frequencies you define.

SCRIPT

Script mode—Allows you to use scripting to link and loop multiple waveforms in complex combinations.

ReferenceClockSource

class nifgen.**ReferenceClockSource**

CLOCK_IN

Specifies that the CLK IN input signal from the front panel connector is used as the Reference Clock source.

NONE

Specifies that a Reference Clock is not used.

ONBOARD_REFERENCE_CLOCK

Specifies that the onboard Reference Clock is used as the Reference Clock source.

PXI_CLOCK

Specifies that the PXI Clock is used as the Reference Clock source.

RTSI_7

Specifies that the RTSI line 7 is used as the Reference Clock source.

RelativeTo

class nifgen.**RelativeTo**

START

CURRENT

SampleClockSource

class nifgen.**SampleClockSource**

CLOCK_IN
Specifies that the signal at the CLK IN front panel connector is used as the Sample Clock source.

DDC_CLOCK_IN
Specifies that the Sample Clock from the DDC connector is used as the Sample Clock source.

ONBOARD_CLOCK
Specifies that the onboard clock is used as the Sample Clock source.

PXI_STAR_LINE
Specifies that the PXI_STAR trigger line is used as the Sample Clock source.

PXI_TRIGGER_LINE_0_RTSI_0
Specifies that the PXI or RTSI line 0 is used as the Sample Clock source.

PXI_TRIGGER_LINE_1_RTSI_1
Specifies that the PXI or RTSI line 1 is used as the Sample Clock source.

PXI_TRIGGER_LINE_2_RTSI_2
Specifies that the PXI or RTSI line 2 is used as the Sample Clock source.

PXI_TRIGGER_LINE_3_RTSI_3
Specifies that the PXI or RTSI line 3 is used as the Sample Clock source.

PXI_TRIGGER_LINE_4_RTSI_4
Specifies that the PXI or RTSI line 4 is used as the Sample Clock source.

PXI_TRIGGER_LINE_5_RTSI_5
Specifies that the PXI or RTSI line 5 is used as the Sample Clock source.

PXI_TRIGGER_LINE_6_RTSI_6
Specifies that the PXI or RTSI line 6 is used as the Sample Clock source.

PXI_TRIGGER_LINE_7_RTSI_7
Specifies that the PXI or RTSI line 7 is used as the Sample Clock source.

SampleClockTimebaseSource

class nifgen.**SampleClockTimebaseSource**

CLOCK_IN

Specifies that the external signal on the CLK IN front panel connector is used as the source.

ONBOARD_CLOCK

Specifies that the onboard Sample Clock timebase is used as the source.

ScriptTriggerDigitalEdgeEdge

class nifgen.**ScriptTriggerDigitalEdgeEdge**

RISING

Rising Edge

FALLING

Falling Edge

ScriptTriggerType

class nifgen.**ScriptTriggerType**

TRIG_NONE

No trigger is configured. Signal generation starts immediately.

DIGITAL_EDGE

Trigger is asserted when a digital edge is detected.

DIGITAL_LEVEL

Trigger is asserted when a digital level is detected.

SOFTWARE_EDGE

Trigger is asserted when a software edge is detected.

StartTriggerDigitalEdgeEdge

class nifgen.**StartTriggerDigitalEdgeEdge**

RISING

Rising Edge

FALLING

Falling Edge

StartTriggerType

class nifgen.**StartTriggerType**

TRIG_NONE
None

DIGITAL_EDGE
Digital Edge

SOFTWARE_EDGE
Software Edge

P2P_ENDPOINT_FULLNESS
P2P Endpoint Fullness

TerminalConfiguration

class nifgen.**TerminalConfiguration**

SINGLE_ENDED
Single-ended operation

DIFFERENTIAL
Differential operation

Trigger

class nifgen.**Trigger**

START

SCRIPT

TriggerMode

class nifgen.**TriggerMode**

SINGLE
Single Trigger Mode - The waveform you describe in the sequence list is generated only once by going through the entire staging list. Only one trigger is required to start the waveform generation. You can use Single trigger mode with the output mode in any mode. After a trigger is received, the waveform generation starts from the first stage and continues through to the last stage. Then, the last stage generates repeatedly until you stop the waveform generation.

CONTINUOUS
Continuous Trigger Mode - The waveform you describe in the staging list generates infinitely by repeatedly cycling through the staging list. After a trigger is received, the waveform generation starts from the first stage and continues through to the last stage. After the last stage completes, the waveform generation loops back to the start of the first stage and continues until it is stopped. Only one trigger is required to start the waveform generation.

STEPPED

Stepped Trigger Mode - After a start trigger is received, the waveform described by the first stage generates. Then, the device waits for the next trigger signal. On the next trigger, the waveform described by the second stage generates, and so on. After the staging list completes, the waveform generation returns to the first stage and continues in a cyclic fashion. After any stage has generated completely, the first eight samples of the next stage are repeated continuously until the next trigger is received. trigger mode.

Note: In Frequency List mode, Stepped trigger mode is the same as Burst

BURST

Burst Trigger Mode - After a start trigger is received, the waveform described by the first stage generates until another trigger is received. At the next trigger, the buffer of the previous stage completes, and then the waveform described by the second stage generates. After the staging list completes, the waveform generation returns to the first stage and continues in a cyclic fashion. In Frequency List mode, the duration instruction is ignored, and the trigger switches the frequency to the next frequency in the list. trigger mode.

Note: In Frequency List mode, Stepped trigger mode is the same as Burst

WaitBehavior

class nifgen.**WaitBehavior**

HOLD_LAST

While in an Idle or Wait state, the output signal remains at the last voltage generated prior to entering the state.

JUMP_TO

While in an Idle or Wait state, the output signal remains at the value configured in the Idle or Wait value property.

Waveform

class nifgen.**Waveform**

SINE

Sinusoid waveform

SQUARE

Square waveform

TRIANGLE

Triange waveform

RAMP_UP

Positive ramp waveform

RAMP_DOWN

Negative ramp waveform

DC

Constant voltage

NOISE

White noise

USER

User-defined waveform as defined by the *nifgen.Session.define_user_standard_waveform()* method.

Exceptions and Warnings

Error

exception nifgen.errors.**Error**

Base exception type that all NI-FGEN exceptions derive from

DriverError

exception nifgen.errors.**DriverError**

An error originating from the NI-FGEN driver

UnsupportedConfigurationError

exception nifgen.errors.**UnsupportedConfigurationError**

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

DriverNotInstalledError

exception nifgen.errors.**DriverNotInstalledError**

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

DriverTooOldError

exception nifgen.errors.**DriverTooOldError**

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

DriverTooNewError

exception nifgen.errors.**DriverTooNewError**

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

InvalidRepeatedCapabilityError

exception nifgen.errors.**InvalidRepeatedCapabilityError**

An error due to an invalid character in a repeated capability

SelfTestError

exception nifgen.errors.**SelfTestError**

An error due to a failed self-test

RpcError

exception nifgen.errors.**RpcError**

An error specific to sessions to the NI gRPC Device Server

DriverWarning

exception nifgen.errors.**DriverWarning**

A warning originating from the NI-FGEN driver

Examples

You can download all nifgen examples here

nifgen_arb_waveform.py

Listing 1: (nifgen_arb_waveform.py)

```
#!/usr/bin/python

import argparse
import math
import nifgen
import sys
import time


def create_waveform_data(number_of_samples):
    waveform_data = []
    angle_per_sample = (2 * math.pi) / number_of_samples
    for i in range(number_of_samples):
        waveform_data.append(math.sin(i * angle_per_sample) * math.sin(i * angle_per_
↪sample * 20))
    return waveform_data


def example(resource_name, options, samples, gain, offset, gen_time):
```

(continues on next page)

(continued from previous page)

```
    waveform_data = create_waveform_data(samples)
    with nifgen.Session(resource_name=resource_name, options=options) as session:
        session.output_mode = nifgen.OutputMode.ARB
        waveform = session.create_waveform(waveform_data_array=waveform_data)
        session.configure_arb_waveform(waveform_handle=waveform, gain=gain, offset=offset)
        with session.initiate():
            time.sleep(gen_time)


def _main(argsv):
    parser = argparse.ArgumentParser(description='Continuously generates an arbitrary waveform.', formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2', help='Resource name of an NI arbitrary waveform generator.')
    parser.add_argument('-s', '--samples', default=100000, type=int, help='Number of samples')
    parser.add_argument('-g', '--gain', default=1.0, type=float, help='Gain')
    parser.add_argument('-o', '--offset', default=0.0, type=float, help='DC offset (V)')
    parser.add_argument('-t', '--time', default=5.0, type=float, help='Generation time (s)')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option string')
    args = parser.parse_args(argsv)
    example(args.resource_name, args.option_string, args.samples, args.gain, args.offset, args.time)


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


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '5433 (2CH)', 'BoardType': 'PXIe', }, }
    example('PXI1Slot2', options, 100000, 1.0, 0.0, 5.0)


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:5433 (2CH);BoardType:PXIe', ]
    _main(cmd_line)


if __name__ == '__main__':
    main()


```

nifgen_script.py

Listing 2: (nifgen_script.py)

```
#!/usr/bin/python

import argparse
import nifgen

import math
import sys
import time

# waveform size should be a multiple of the quantum, which is 4, 2 or 1, for all devices
# minimum waveform size needed to prevent underflow varies with sample rate and device.
# If underflow occurs, increase this value.
NUMBER_OF_SAMPLES = 2096


# waveforms finish just short of 360 degrees, so that we don't repeat the first point
# if we repeat the waveform
SINE_WAVE = [math.sin(math.pi * 2 * x / (NUMBER_OF_SAMPLES)) for x in range(NUMBER_OF_
↪SAMPLES)]
RAMP_UP = [x / (NUMBER_OF_SAMPLES) for x in range(NUMBER_OF_SAMPLES)]
RAMP_DOWN = [-1.0 * x for x in RAMP_UP]
SQUARE_WAVE = [1.0 if x < (NUMBER_OF_SAMPLES / 2) else -1.0 for x in range(NUMBER_OF_
↪SAMPLES)]
SAWTOOTH_WAVE = RAMP_UP[::2] + [(-1 + x) for x in RAMP_UP][::2]


SCRIPT_ALL = '''
script scriptmulti
  repeat until scriptTrigger0
    generate rampup
    generate sine
    generate rampdown
  end repeat
  repeat until scriptTrigger0
    generate rampdown
    generate square
    generate rampup
  end repeat
  repeat until scriptTrigger0
    generate rampup
    generate rampdown
  end repeat
  repeat until scriptTrigger0
    generate sine
  end repeat
  repeat until scriptTrigger0
    generate sawtooth
  end repeat
  repeat until scriptTrigger0
```

(continues on next page)

(continued from previous page)

```
    generate rampdown
    generate rampup
  end repeat
end script

script scriptsine
  repeat until scriptTrigger0
    generate sine
  end repeat
end script

script scriptrampup
  repeat until scriptTrigger0
    generate rampup
  end repeat
end script

script scriptrampdown
  repeat until scriptTrigger0
    generate rampdown
  end repeat
end script

script scriptsquare
  repeat until scriptTrigger0
    generate square
  end repeat
end script

script scriptsawtooth
  repeat until scriptTrigger0
    generate sawtooth
  end repeat
end script
'''


def example(resource_name, options, shape, channel):
    with nifgen.Session(resource_name=resource_name, options=options, channel_
↪name=channel) as session:
        # CONFIGURATION
        # 1 - Set the mode to Script
        session.output_mode = nifgen.OutputMode.SCRIPT

        # 2 - Configure Trigger:
        # SOFTWARE TRIGGER: used in the script
        session.script_triggers[0].script_trigger_type = nifgen.ScriptTriggerType.
↪SOFTWARE_EDGE  # TRIG_NONE / DIGITAL_EDGE / DIGITAL_LEVEL / SOFTWARE_EDGE
        session.script_triggers[0].digital_edge_script_trigger_edge = nifgen.
↪ScriptTriggerDigitalEdgeEdge.RISING  # RISING / FAILING

        # 3 - Calculate and write different waveform data to the device's onboard memory
```

(continues on next page)

(continued from previous page)

```
        session.channels[channel].write_waveform('sine', SINE_WAVE)         # (waveform_
↪name, data)
        session.channels[channel].write_waveform('rampup', RAMP_UP)
        session.channels[channel].write_waveform('rampdown', RAMP_DOWN)
        session.channels[channel].write_waveform('square', SQUARE_WAVE)
        session.channels[channel].write_waveform('sawtooth', SAWTOOTH_WAVE)

        # 4 - Script to generate
        # supported shapes: SINE / SQUARE / SAWTOOTH / RAMPUP / RAMPDOWN / MULTI
        script_name = 'script{}'.format(shape.lower())
        num_triggers = 6 if shape.upper() == 'MULTI' else 1  # Only multi needs multiple␣
↪triggers, all others need one

        session.channels[channel].write_script(SCRIPT_ALL)
        session.script_to_generate = script_name

        # LAUNCH
        with session.initiate():
            for x in range(num_triggers):
                time.sleep(10)
                session.script_triggers[0].send_software_edge_trigger()


def _main(argsv):
    parser = argparse.ArgumentParser(description='Generate different shape waveforms.',␣
↪formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2', help='Resource␣
↪name of an NI arbitrary waveform generator.')
    parser.add_argument('-s', '--shape', default='SINE', help='Shape of the signal to␣
↪generate')
    parser.add_argument('-c', '--channel', default='0', help='Channel to use when␣
↪generating')
    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.shape.upper(), args.channel)


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '5433 (2CH)', 'BoardType':
↪'PXIe', }, }
    example('PXI1Slot2', options, 'SINE', '0')


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:5433 (2CH);
↪BoardType:PXIe', '--channel', '0', ]
    _main(cmd_line)


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

(continues on next page)

(continued from previous page)

```


if __name__ == '__main__':
    main()



```

nifgen_standard_function.py

Listing 3: (nifgen_standard_function.py)

```
#!/usr/bin/python

import argparse
import nifgen
import sys
import time


def example(resource_name, options, waveform, frequency, amplitude, offset, phase, gen_time):
    with nifgen.Session(resource_name=resource_name, options=options) as session:
        session.output_mode = nifgen.OutputMode.FUNC
        session.configure_standard_waveform(waveform=nifgen.Waveform[waveform], amplitude=amplitude, frequency=frequency, dc_offset=offset, start_phase=phase)
        with session.initiate():
            time.sleep(gen_time)


def _main(argsv):
    supported_waveforms = list(nifgen.Waveform.__members__.keys())[:-1]  # no support for user-defined waveforms in example
    parser = argparse.ArgumentParser(description='Generates the standard function.', formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n', '--resource-name', default='PXI1Slot2', help='Resource name of an NI function generator.')
    parser.add_argument('-w', '--waveform', default=supported_waveforms[0], choices=supported_waveforms, type=str.upper, help='Standard waveform')
    parser.add_argument('-f', '--frequency', default=1000, type=float, help='Frequency (Hz)')
    parser.add_argument('-a', '--amplitude', default=1.0, type=float, help='Amplitude (Vpk-pk)')
    parser.add_argument('-o', '--offset', default=0.0, type=float, help='DC offset (V)')
    parser.add_argument('-p', '--phase', default=0.0, type=float, help='Start phase (deg)')
    parser.add_argument('-t', '--time', default=5.0, type=float, help='Generation time (s)')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option
```

(continues on next page)

(continued from previous page)

```
↪string')
    args = parser.parse_args(argsv)
    example(args.resource_name, args.option_string, args.waveform, args.frequency, args.
↪amplitude, args.offset, args.phase, args.time)


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


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '5433 (2CH)', 'BoardType':
↪'PXIe', }, }
    example('PXI1Slot2', options, 'SINE', 1000, 1.0, 0.0, 0.0, 5.0)


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:5433 (2CH);
↪BoardType:PXIe', ]
    _main(cmd_line)


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

nifgen_trigger.py

Listing 4: (nifgen_trigger.py)

```
import argparse
import nifgen
import sys
import time


def example(resource_name1, resource_name2, options, waveform, gen_time):
    with nifgen.Session(resource_name=resource_name1, options=options) as session1,␣
↪nifgen.Session(resource_name=resource_name2, options=options) as session2:
        session_list = [session1, session2]
        for session in session_list:
            session.output_mode = nifgen.OutputMode.FUNC
            session.configure_standard_waveform(waveform=nifgen.Waveform[waveform],␣
↪amplitude=1.0, frequency=1000, dc_offset=0.0, start_phase=0.0)
        session1.start_trigger_type = nifgen.StartTriggerType.SOFTWARE_EDGE
        session2.start_trigger_type = nifgen.StartTriggerType.DIGITAL_EDGE
        session2.digital_edge_start_trigger_edge = nifgen.StartTriggerDigitalEdgeEdge.
↪RISING
```

(continues on next page)

(continued from previous page)

```
        session2.digital_edge_start_trigger_source = '/' + resource_name1 + '/0/
↪StartTrigger'
        with session2.initiate():
            with session1.initiate():
                session1.send_software_edge_trigger(nifgen.Trigger.START)
                time.sleep(gen_time)


def _main(argsv):
    supported_waveforms = list(nifgen.Waveform.__members__.keys())[:-1]  # no support␣
↪for user-defined waveforms in example
    parser = argparse.ArgumentParser(description='Triggers one device on the start␣
↪trigger of another device.', formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n1', '--resource-name1', default='PXI1Slot2', help='Resource␣
↪name of an NI function generator.')
    parser.add_argument('-n2', '--resource-name2', default='PXI1Slot3', help='Resource␣
↪name of an NI function generator.')
    parser.add_argument('-w', '--waveform', default=supported_waveforms[0],␣
↪choices=supported_waveforms, type=str.upper, help='Standard waveform')
    parser.add_argument('-t', '--time', default=5.0, type=float, help='Generation time␣
↪(s)')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option␣
↪string')
    args = parser.parse_args(argsv)
    example(args.resource_name1, args.resource_name2, args.option_string, args.waveform,␣
↪args.time)


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


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '5433 (2CH)', 'BoardType':
↪'PXIe', }, }
    example('PXI1Slot2', 'PXI1Slot3', options, 'SINE', 5.0)


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:5433 (2CH);
↪BoardType:PXIe', ]
    _main(cmd_line)


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

gRPC Support

Support for using NI-FGEN over gRPC

SessionInitializationBehavior

class nifgen.**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 nifgen.**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** (*nifgen.SessionInitializationBehavior*) – Specifies whether it is acceptable to initialize a new session or attach to an existing one, or if only one of the behaviors is desired.

  The driver session exists on the NI gRPC Device Server.

4.2 Additional Documentation

Refer to your driver documentation for device-specific information and detailed API documentation.

Refer to the nimi-python Read the Docs project for documentation of versions 1.4.4 of the module or earlier.

CHAPTER

FIVE

LICENSE

nimi-python is licensed under an MIT-style license (see LICENSE). Other incorporated projects may be licensed under different licenses. All licenses allow for non-commercial and commercial use.

gRPC Features

For driver APIs that support it, passing a GrpcSessionOptions instance as a parameter to Session.__init__() is subject to the NI General Purpose EULA (see NILICENSE).

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

E

F

G

H

I

J

L

M

N

O

T

U

V

W