

niDCPower_ConfigureOutputEnabled

ViStatus niDCPower_ConfigureOutputEnabled (ViSession vi, ViConstString
channelName, ViBoolean enabled);

Purpose

Enables or disables generation on the specified channel(s). Depending on the selected output function, the voltage level or the current level must be set in addition to enabling the output to generate the desired level. Refer to the [niDCPower_ConfigureVoltageLevel](#) function, the [niDCPower_ConfigureCurrentLevel](#) function, and the [niDCPower_ConfigureOutputFunction](#) function for more information about configuring the desired output level.



Note If the device is in [Delayed Configuration mode](#), enabling the output will not take effect until you call the [niDCPower_Initiate](#) function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
enabled	ViBoolean	Specifies whether the output is enabled or disabled.

Defined Values:

VI_TRUE	Enables generation on the specified output channel(s).
VI_FALSE	Disables generation on the specified output channel(s).

Default Value: VI_TRUE

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureOutputFunction

ViStatus niDCPower_ConfigureOutputFunction (ViSession vi, ViConstString
channelName, ViBoolean function);

Purpose

Configures the function the device attempts to generate for the specified channel(s).

When `NIDCPOWER_VAL_DC_VOLTAGE` is selected, the device generates the desired voltage level on the output as long as the output current is below the current limit. The following functions can be used to configure the channel when `NIDCPOWER_VAL_DC_VOLTAGE` is selected:

- [`niDCPower_ConfigureVoltageLevel`](#)
- [`niDCPower_ConfigureCurrentLimit`](#)
- [`niDCPower_ConfigureVoltageLevelRange`](#)
- [`niDCPower_ConfigureCurrentLimitRange`](#)

When `NIDCPOWER_VAL_DC_CURRENT` is selected, the device generates the desired current level on the output as long as the output voltage is below the voltage limit. The following functions can be used to configure the channel when `NIDCPOWER_VAL_DC_CURRENT` is selected:

- [`niDCPower_ConfigureCurrentLevel`](#)
- [`niDCPower_ConfigureVoltageLimit`](#)
- [`niDCPower_ConfigureCurrentLevelRange`](#)
- [`niDCPower_ConfigureVoltageLimitRange`](#)

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
function	ViInt32	Configures the function to generate for the specified channel(s).

Defined Values:

NIDCPOWER_VAL_DC_VOLTAGE	Sets the output function to DC voltage.
NIDCPOWER_VAL_DC_CURRENT	Sets the output function to DC current.

Default Value: NIDCPOWER_VAL_DC_VOLTAGE

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_MeasureMultiple

```
ViStatus niDCPower_MeasureMultiple (ViSession vi, ViConstString  
    channelName, ViReal64 voltageMeasurements[], ViReal64  
    currentMeasurements[]);
```

Purpose

Returns arrays of the measured voltage and current values on the specified output channel(s). Each call to this function blocks other function calls until the measurements are returned from the device. The order of the measurements returned in the array corresponds to the order on the specified output channel(s). The measurement speed of the device and the [NIDCPOWER_ATTR_SAMPLES_TO_AVERAGE](#) attribute dictates the length of time that a measurement takes.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies the output channels to measure. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2).

Output

Name	Type	Description
voltageMeasurements	ViReal64[]	Returns an array of voltage measurements. The measurements in the array are returned in the same order as the channels specified in channelName . Ensure that sufficient space has been allocated for the returned array.
currentMeasurements	ViReal64[]	Returns an array of current measurements. The measurements in the array are returned in the same order as

the channels specified in **channelName**. Ensure that sufficient space has been allocated for the returned array.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureSense

ViStatus niDCPower_ConfigureSense (ViSession vi, ViConstString
channelName, ViBoolean sense);

Purpose

Specifies whether to use [local](#) or [remote](#) sensing of the output voltage on the specified channel(s). Refer to the *Devices* topic specific to your device in the *NI DC Power Supplies and SMUs Help* to find out more information about sensing on supported channels.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
sense	ViInt32	Specifies local or remote sensing on the specified channel(s).

Defined Values:

NIDCPOWER_VAL_LOCAL	Local sensing
NIDCPOWER_VAL_REMOTE	Remote sensing

Default Value:

NIDCPOWER_VAL_LOCAL

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_Initiate

ViStatus niDCPower_Initiate (ViSession vi);

Purpose

Commits the configured settings to hardware and places the device in [Immediate mode](#). Any configuration calls made after this function are applied immediately. To commit simultaneous hardware settings on multiple output channels, call the [niDCPower_Abort](#) function, configure the power supply, and call this function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_Abort

ViStatus niDCPower_Abort (ViSession vi);

Purpose

Places the device in [Delayed Configuration mode](#). Any configuration functions called after this function are not applied until the [niDCPower_Initiate](#) function is called. If power output is enabled when you call the `niDCPower_Abort` function, the output channels remain in their current state and continue providing power.

Use the [niDCPower_ConfigureOutputEnabled](#) function to disable power output on a per channel basis. Use the [niDCPower_reset](#) function to disable output on all channels. While in Delayed Configuration mode, NI-DCPower performs only generic parameter validation. Any conflicting configuration calls are not validated until the `niDCPower_Initiate` function is called. If the same configuration is set multiple times to different values, NI-DCPower uses the last configuration call.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureVoltageLevel

ViStatus niDCPower_ConfigureVoltageLevel (ViSession vi, ViConstString
channelName, ViReal64 level);

Purpose

Configures the voltage level the device attempts to generate for the specified channel(s). The channel must be enabled for the specified voltage level to take effect. Refer to the [niDCPower_ConfigureOutputEnabled](#) attribute for more information about enabling the output channel.

The voltage level setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_VOLTAGE output function using the [niDCPower_ConfigureOutputFunction](#) function. The device actively regulates the voltage at the specified level unless doing so causes a current output greater than the [current limit](#) through the channels' output terminals.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
level	ViReal64	Specifies the voltage level, in volts, for the output channel generation. Valid Values: The valid values for this parameter are defined by the voltage level range that is selected using the niDCPower_ConfigureVoltageLevelRange function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureVoltageLevelRange

```
ViStatus niDCPower_ConfigureVoltageLevel Range(ViSession vi,  
ViConstString channelName, ViReal64 range);
```

Purpose

Configures the voltage level range for the specified channel(s). The configured range defines the valid values the voltage level can be set to using the [niDCPower_ConfigureVoltageLevel](#) function. The voltage level range setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_VOLTAGE output function using the [niDCPower_ConfigureOutputFunction](#) function.

Use the [NIDCPOWER_ATTR_VOLTAGE_LEVEL_AUTORANGE](#) attribute to enable automatic selection of the voltage level range.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
range	ViReal64	Specifies the voltage level range, in volts, on the specified channel(s).

Valid Values:

NI PXI-4110

Channel	Voltage Level Range (V)	Voltage Level (V)
0	6	0 to +6
1	20	0 to +20
2	20	0 to -20

NI PXI-4130

--	--	--

Channel	Voltage Level Range (V)	Voltage Level (V)
0	6	0 to +6
1	6	-6 to +6
	20	-20 to +20



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest range. For example, requesting the 10 V voltage level range on Channel 1 on the NI-PXI 4130 coerces the voltage level range to 20 V. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureCurrentLimit

ViStatus niDCPower_ConfigureCurrentLimit (ViSession vi, ViConstString
channelName, ViInt32 behavior, ViReal64 limit);

Purpose

Configures the current limit for the specified channel(s). The channel must be enabled for the specified current limit to take effect. Refer to the [niDCPower_ConfigureOutputEnabled](#) function for more information about enabling the output channel.

The current limit is the current that the output should not exceed when generating the desired [voltage level](#). The current limit setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_VOLTAGE output function using the [niDCPower_ConfigureOutputFunction](#) function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel 0-2 specifies channels 0, 1, and 2). In immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
behavior	ViInt32	Specifies how the output should behave when the current limit is reached

Defined Values:

NIDCPOWER_VAL_CURRENT_REGULATE	Control output current so that does not exceed the current limit. Power continues to generate even if the current limit is reached
--------------------------------	--

limit	ViReal64	Specifies the current limit, in amps, on the specified channel(s).
--------------	----------	--

Valid Values:

The valid values for this parameter are defined by the current limit range 1 is configured using the [niDCPower_ConfigureCurrentLimitRange](#) function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureCurrentLimitRange

```
ViStatus niDCPower_ConfigureCurrentLimit Range(ViSession vi,  
        ViConstString channelName, ViReal64 range);
```

Purpose

Configures the current limit range for the specified channel(s). The configured range defines the valid values the current limit can be set to using the [niDCPower_ConfigureCurrentLimit](#) function. The current limit range setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_VOLTAGE output function using the [niDCPower_ConfigureOutputFunction](#) function.

Use the [NIDCPOWER_ATTR_CURRENT_LIMIT_AUTORANGE](#) attribute to enable automatic selection of the current limit range.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
range	ViReal64	Specifies the current limit range, in amps, for the specified channel.

Valid Values:

NI PXI-4110

Channel	Current Limit Range	Current Limit
0	1 A	+0.01 to +1 A
1, 2	20 mA	+0.20 to +20 mA
	1 A	+0.01 to +1 A

NI PXI-4130

Channel	Current Limit Range	Current Limit
0	1 A	+0.02 to +1 A
1	200 μ A	+4 to +200 μ A
	2 mA	+0.04 to +2 mA
	20 mA	+0.40 to +20 mA
	200 mA	+4 to +200 mA
	2 A	+0.04 to +2 A



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest range. For example, requesting the 100 mA current limit range on Channel 1 on the NI-PXI 4130 coerces the current level range to 200 mA. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureCurrentLevel

ViStatus niDCPower_ConfigureCurrentLevel (ViSession vi, ViConstString
channelName, ViReal64 level);

Purpose

Configures the current level the device attempts to generate for the specified channel(s). The channel must be enabled for the specified current level to take effect. Refer to the [niDCPower_ConfigureOutputEnabled](#) function for more information about enabling the output channel.

The current level setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_CURRENT output function using the [niDCPower_ConfigureOutputFunction](#) function. The device actively regulates the current at the specified level unless doing so causes a voltage greater than the [voltage limit](#) across the channels' output terminals.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
level	ViReal64	Specifies the current level, in amps, to generate for the specified channel(s).

Valid Values:

The valid values for this parameter are defined by the current level range that is configured using the [niDCPower_ConfigureCurrentLevelRange](#) function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureCurrentLevelRange

ViStatus niDCPower_ConfigureCurrentLevelRange (ViSession vi,
ViConstString channelName, ViReal64 range);

Purpose

Configures the current level range for the specified channel(s). The configured range defines the valid values the current level can be set to using the [niDCPower_ConfigureCurrentLevel](#) function. The current level range setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_CURRENT output function using the [niDCPower_ConfigureOutputFunction](#) function.

Use the [NIDCPOWER_ATTR_CURRENT_LEVEL_AUTORANGE](#) attribute to enable automatic selection of the current level range.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
range	ViReal64	Specifies the current level range, in amps, for the specified channel.

Valid Values:

NI PXI-4110

Channel	Current Level Range	Current Level
0	1 A	+0.01 to +1 A
1	20 mA	+0.20 to +20 mA
	1 A	+0.01 to +1 A

2	20 mA	-0.20 to -20 mA
	1 A	-0.01 to -1 A

NI PXI-4130

Channel	Current Level Range	Current Level
0	1 A	+0.02 to +1 A
1	200 μ A	+4 to +200 μ A and -4 to -200 μ A
	2 mA	+0.04 to +2 mA and -0.04 to -2 mA
	20 mA	+0.40 to +20 mA and -0.40 to -20 mA
	200 mA	+4 to +200 mA and -4 to -200 mA
	2 A	+0.04 to +2 A and -0.04 to -2 A



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest range. For example, requesting the 100 mA current level range on Channel 1 on the NI-PXI 4130 coerces the current level range to 200 mA. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureVoltageLimit

ViStatus niDCPower_ConfigureVoltageLimit (ViSession vi, ViConstString
channelName, ViReal64 limit);

Purpose

Configures the voltage limit for the specified channel(s). The channel must be enabled for the specified voltage limit to take effect. Refer to the [niDCPower_ConfigureOutputEnabled](#) function for more information about enabling the output channel.

The voltage limit is the voltage that the output should not exceed when generating the desired [current level](#). The voltage limit setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_CURRENT output function using the [niDCPower_ConfigureOutputFunction](#) function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
limit	ViReal64	Specifies the voltage limit, in volts, on the specified output channel(s).

Valid Values:
The valid values for this parameter are defined by the voltage limit range that is configured using the [niDCPower_ConfigureVoltageLimitRange](#) function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ConfigureVoltageLimitRange

ViStatus niDCPower_ConfigureVoltageLimitRange (ViSession vi,
ViConstString channelName, ViReal64 range);

Purpose

Configures the voltage limit range for the specified channel(s). The configured range defines the valid values the voltage limit can be set to using the [niDCPower_ConfigureVoltageLimit](#) function. The voltage limit range setting is applicable only if the channel is set to the NIDCPOWER_VAL_DC_CURRENT output function using the [niDCPower_ConfigureOutputFunction](#) function.

Use the [NIDCPOWER_ATTR_VOLTAGE_LIMIT_AUTORANGE](#) attribute to enable automatic selection of the voltage limit range.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies to which output channel(s) to apply this configuration value. You can specify multiple channels by using a channel list or a channel range. A channel list is a comma (,) separated sequence of channel names (e.g. 0,2 specifies channels 0 and 2). A channel range is a lower bound channel followed by a hyphen (-) or colon (:) followed by an upper bound channel (e.g. 0-2 specifies channels 0, 1, and 2). In Immediate mode , multiple output channel configurations are performed sequentially based on the order specified in this parameter.
range	ViReal64	Specifies the voltage limit range, in volts, on the specified channel(s).

Valid Values:

NI PXI-4110

Channel	Voltage Limit Range (V)	Voltage Limit (V)
0	6	0 to +6
1, 2	20	0 to +20

NI PXI-4130

Channel	Voltage Limit Range (V)	Voltage Limit (V)
---------	-------------------------	-------------------

0	6	0 to +6
1	6	0 to +6
	20	0 to +20



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest range. For example, requesting the 10 V voltage limit range on Channel 1 on the NI-PXI 4130 coerces the voltage limit range to 20 V. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_SetAttributeViInt32

ViStatus niDCPower_SetAttributeViInt32(ViSession vi, ViConstString
channelName, ViAttr attribute, ViInt32 value)

Purpose

Sets the value of a VInt32 attribute.

This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

NI-DCPower contains high-level functions that set most of the device attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, this function checks the device status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform device I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant device I/O.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViInt32. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.
value	ViInt32	Specifies the value to which you want to set the attribute. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .



Note Some of the values might not be valid depending upon the current settings of the device session.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_SetAttributeViReal64

ViStatus niDCPower_SetAttributeViReal64(ViSession vi, ViConstString
channelName, ViAttr attribute, ViReal64 value)

Purpose

Sets the value of a ViReal64 attribute.

This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

NI-DCPower contains high-level functions that set most of the device attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, this function checks the device status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform device I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant device I/O.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViReal64. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.
value	ViReal64	Specifies the value to which you want to set the attribute. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .



Note Some of the values might not be valid depending upon the current settings of the device session.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_SetAttributeViString

ViStatus niDCPower_SetAttributeViString(ViSession vi, ViConstString
channelName, ViAttr attribute, ViConstString value)

Purpose

Sets the value of a ViString attribute.

This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

NI-DCPower contains high-level functions that set most of the device attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, this function checks the device status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform device I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant device I/O.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViString. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.
value	ViConstString	Specifies the value to which you want to set the attribute. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .



Note Some of the values might not be valid depending upon the current settings of the device session.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_SetAttributeViBoolean

ViStatus niDCPower_SetAttributeViBoolean(ViSession vi, ViConstString
channelName, ViAttr attribute, ViBoolean value)

Purpose

Sets the value of a ViBoolean attribute.

This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

NI-DCPower contains high-level functions that set most of the device attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, this function checks the device status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform device I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant device I/O.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViBoolean. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.
value	ViBoolean	Specifies the value to which you want to set the attribute. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .



Note Some of the values might not be valid depending upon the current settings of the device session.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_SetAttributeViSession

ViStatus niDCPower_SetAttributeViSession(ViSession vi, ViConstString
channelName, ViAttr attribute, ViSession value)

Purpose

Sets the value of a ViSession attribute.

This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

NI-DCPower contains high-level functions that set most of the device attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, this function checks the device status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform device I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant device I/O.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViSession. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.
value	ViSession	Specifies the value to which you want to set the attribute. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .



Note Some of the values might not be valid depending upon the current settings of the device session.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetAttributeViInt32

ViStatus niDCPower_GetAttributeViInt32(ViSession vi, ViConstString
channelName, ViAttr attribute, ViInt32 *value)

Purpose

Queries the value of a ViInt32 attribute.

You can use this function to get the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to get the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel name is 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViInt32. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.

Output

Name	Type	Description
value	ViInt32*	Returns the current value of the attribute. Passes the address of a ViInt32 variable. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetAttributeViReal64

ViStatus niDCPower_GetAttributeViReal64(ViSession vi, ViConstString
channelName, ViAttr attribute, ViReal64 *value)

Purpose

Queries the value of a ViReal64 attribute.

You can use this function to get the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViReal64. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.

Output

Name	Type	Description
value	ViReal64*	Returns the current value of the attribute. Passes the address of a ViReal64 variable. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetAttributeViString

ViStatus niDCPower_GetAttributeViString(ViSession vi, ViConstString
channelName, ViAttr attribute, ViInt32 bufferSize, ViChar value[])

Purpose

Queries the value of a ViString attribute.

You can use this function to get the values of device-specific attributes and inherent IVI attributes. If the attribute represents an device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViString. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.
bufferSize	ViInt32	Passes the number of bytes in the buffer and specifies the number of bytes in the ViChar array you specify for value . If the current value of value , including the terminating NUL byte, is larger than the size you indicate in this attribute, the function copies (buffer size - 1) bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is 123456 and the buffer size is 4, the function places 123 into the buffer and returns 7. To obtain the required buffer size, you can pass 0 for this attribute and VI_NULL for value . If you want the function to fill in the buffer regardless of the number of bytes in the value, pass a negative number for this attribute.

Output

Name	Type	Description
value	ViChar[]	The buffer in which the function returns the current value of the attribute. The buffer must be of type ViChar and have at least as many bytes as indicated in bufferSize . If the current value of the attribute, including the terminating NUL byte,

contains more bytes than you indicate in this attribute, the function copies (buffer size -1) bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is 123456 and the buffer size is 4, the function places 123 into the buffer and returns 7.

If you specify 0 for **bufferSize**, you can pass VI_NULL for this attribute.

If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing **Enter** on this control. Select a value by double-clicking on it or by selecting it and then pressing **Enter**.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call niDCPower_error_message . To obtain additional information concerning the error condition, call niDCPower_GetError .
---------------	----------	--

niDCPower_GetAttributeViBoolean

ViStatus niDCPower_GetAttributeViBoolean(ViSession vi, ViConstString
channelName, ViAttr attribute, ViBoolean *value)

Purpose

Queries the value of a ViBoolean attribute.

You can use this function to get the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViBoolean. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.

Output

Name	Type	Description
value	ViBoolean*	Returns the current value of the attribute. Passes the address of a ViBoolean variable. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetAttributeViSession

ViStatus niDCPower_GetAttributeViSession(ViSession vi, ViConstString
channelName, ViAttr attribute, ViSession *value)

Purpose

Queries the value of a ViSession attribute.

You can use this function to get the values of device-specific attributes and inherent IVI attributes. If the attribute represents a device state, this function performs device I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	If the attribute is channel-based, this attribute specifies the name of the channel on which to set the value of the attribute. If the attribute is not channel-based, then pass VI_NULL or an empty string. Valid channel names are 0, 1, and 2.
attribute	ViAttr	Specifies the ID of an attribute. From the function panel window, you can use this control as follows. <ul style="list-style-type: none">• In the function panel window, click on the control or press Enter or the spacebar to display a dialog box containing hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing Enter.• A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of type ViSession. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.• If you want to enter a variable name, press Ctrl+T to change this ring control to a manual input box. If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the value control and pressing Enter.

Output

Name	Type	Description
value	ViSession*	Returns the current value of the attribute. Passes the address of a ViSession variable. If the attribute currently showing in the attribute ring control has constants as valid values, you can view a list of the constants by pressing Enter on this control. Select a value by double-clicking on it or by selecting it and then pressing Enter .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_QueryOutputState

```
ViStatus niDCPower_QueryOutputState (ViSession vi, ViConstString  
channelName, ViInt32 outputState, ViBoolean *inState);
```

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower niDCPower InitWithOptions function.
channelName	ViConstString	Specifies the output channel to query. The output state may only be queried at a time.
outputState	ViInt32	Specifies the outputState of the output channel that is being queried.

Defined Values:

NIDCPOWER_VAL_OUTPUT_CONSTANT_VOLTAGE

NIDCPOWER_VAL_OUTPUT_CONSTANT_CURRENT

Default Value: NIDCPOWER_VAL_OUTPUT_CONSTANT_VOLTAGE

Output

Name	Type	Description
inState	ViBoolean*	Returns whether the device output channel is in the specified state.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_QueryInCompliance

```
ViStatus niDCPower_QueryInCompliance(ViSession vi, ViConstString  
channelName, ViBoolean *inCompliance);
```

Purpose

Queries the device to indicate if the output is operating at the compliance limit.

The compliance limit is the current limit when the output function is set to `NIDCPOWER_VAL_DC_VOLTAGE`. If the output is operating at the compliance limit, the output reaches the current limit before the desired voltage level. Refer to the [niDCPower_ConfigureOutputFunction](#) function and the [niDCPower_ConfigureCurrentLimit](#) function for more information about output function and current limit, respectively.

The compliance limit is the voltage limit when the output function is set to `NIDCPOWER_VAL_DC_CURRENT`. If the output is operating at the compliance limit, the output reaches the voltage limit before the desired current level. Refer to the `niDCPower_ConfigureOutputFunction` function and the [niDCPower_ConfigureVoltageLimit](#) function for more information about output function and voltage limit, respectively.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies the output channel to query. Compliance status can only be queried for one channel at a time.

Output

Name	Type	Description
inCompliance	ViBoolean*	Returns whether the device output channel is in compliance.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_Measure

ViStatus niDCPower_Measure (ViSession vi, ViConstString channelName,
ViInt32 measurementType, ViReal64 *measurement)

Purpose

Returns the measured value of either the voltage or current on the specified output channel. Each call to this function blocks other function calls until the hardware returns the **measurement**. The **measurement** speed of the device and the [NIDCPOWER_ATTR_SAMPLES_TO_AVERAGE](#) attribute dictate the length of time that a **measurement** takes. To measure multiple output channels, use the [niDCPower_MeasureMultiple](#) function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies the output channel to measure. The channel can be measured at a time with the niDCPower_MeasureMultiple function to measure multiple channels.
measurementType	ViInt32	Specifies whether a voltage or current is to be measured.

Defined Values:

NIDCPOWER_VAL_MEASURE_VOLTAGE

NIDCPOWER_VAL_MEASURE_CURRENT

Default Value:

NIDCPOWER_VAL_MEASURE_VOLTAGE

Output

Name	Type	Description
measurement	ViReal64*	Returns the value of the measurement for voltage or current.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_QueryMaxCurrentLimit

ViStatus niDCPower_QueryMaxCurrentLimit (ViSession vi, ViConstString
channelName, ViReal64 voltageLevel, ViReal64 *maxCurrentLimit);

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies the output channel to query. The maximum current limit may only be queried for one channel at a time.
voltageLevel	ViReal64	Specifies the voltage level to use when calculating the maxCurrentLimit .

Output

Name	Type	Description
maxCurrentLimit	ViReal64*	Returns the maximum current limit that can be set with the specified voltageLevel .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_QueryMaxVoltageLevel

ViStatus niDCPower_QueryMaxVoltageLevel (ViSession vi, ViConstString
channelName, ViReal64 currentLimit, ViReal64 *maxVoltageLevel);

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies the output channel to query. The maximum voltage level may only be queried for one channel at a time.
currentLimit	ViReal64	Specifies the current limit to use when calculating the maxVoltageLevel .

Output

Name	Type	Description
maxVoltageLevel	ViReal64*	Returns the maximum voltage level that can be set on an output channel with the specified currentLimit .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_QueryMinCurrentLimit

ViStatus niDCPower_QueryMinCurrentLimit (ViSession vi, ViConstString
channelName, ViReal64 voltageLevel, ViReal64 *minCurrentLimit);

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies the output channel to query. The minimum current limit may only be queried for one channel at a time.
voltageLevel	ViReal64	Specifies the voltage level to use when calculating the minCurrentLimit .

Output

Name	Type	Description
minCurrentLimit	ViReal64*	Returns the minimum current limit that can be set on an output channel with the specified voltageLevel .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_InitExtCal

```
ViStatus niDCPower_InitExtCal (ViRsrc resourceName, ViConstString  
    password, ViSession *newVi);
```

Purpose

If **password** is valid, this function creates a new IVI instrument driver session to the device specified in **resourceName** and returns an instrument handle you use to identify the device in all subsequent NI-DCPower function calls. This function also sends initialization commands to set the device to the state necessary for the operation of NI-DCPower.

Opening a calibration session always performs a reset. Refer to the calibration procedure for the device you are calibrating for detailed instructions on the appropriate use of this function.

Parameters

Input

Name	Type	Description
resourceName	ViRsrc	Specifies the resourceName assigned by Measurement & Automation Explorer (MAX), for example "PXI1Slot3" where "PXI1Slot3" is an instrument's resourceName . resourceName can also be a logical IVI name.
password	ViConstString	Specifies the password for opening a calibration session. The initial password is factory configured to "NI". password can be a maximum of four alphanumeric characters.

Output

Name	Type	Description
newVi	ViSession*	Returns a handle that you use to identify the session in all subsequent NI-DCPower function calls.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_CloseExtCal

ViStatus niDCPower_CloseExtCal (ViSession vi, ViInt32 action);

Purpose

Closes the session specified in **vi** and deallocates the resources that NI-DCPower reserved for calibration. Refer to the calibration procedure for the device you are calibrating for detailed instructions on the appropriate use of this function. If an error occurs before this function, **action** defaults to NIDCPOWER_VAL_CANCEL.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower InitExtCal function.
action	ViInt32	Specifies how to use the calibration values from this session as the session is closed.

Defined Values:

NIDCPOWER_VAL_COMMIT	The new calibration constants are stored in the EEPROM.
NIDCPOWER_VAL_CANCEL	The old calibration constants are kept, and the new ones are discarded.

Default Value: NIDCPOWER_VAL_CANCEL

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_CalAdjustVoltageLevel

ViStatus niDCPower_CalAdjustVoltageLevel (ViSession vi, ViConstString
channelName, ViReal64 range, ViUInt32 numberOfMeasurements,
ViReal64 requestedOutputs[], ViReal64 measuredOutputs[]);

Purpose

Calculates the calibration constants for the voltage level for the specified output channel. This function compares the array in **requestedOutputs** to the array in **measuredOutputs** and calculates the calibration constants for the voltage level of the output channel. Refer to the calibration procedure of the device you are calibrating for detailed instructions on the appropriate use of this function. This function can only be called in an external calibration session.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower_InitExtCal function.
channelName	ViConstString	Specifies the output channel to which these calibration settings apply.
range	ViReal64	Specifies the range to calibrate with these settings. Only one channel at a time may be calibrated.
numberOfMeasurements	ViUInt32	Specifies the number of elements in requestedOutputs and measuredOutputs .
requestedOutputs	ViReal64[]	Specifies an array of the output values requested in the niDCPower_ConfigureVoltageLevel function.
measuredOutputs	ViReal64[]	Specifies an array of the output values measured by an external precision digital multimeter.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_CalAdjustVoltageMeasurement

```
ViStatus niDCPower_CalAdjustVoltageMeasurement (ViSession vi,  
ViConstString channelName, ViReal64 range, ViUInt32  
numberOfMeasurements, ViReal64 reportedOutputs[], ViReal64  
measuredOutputs[]);
```

Purpose

Calculates the calibration constants for the voltage measurements returned by the [niDCPower_Measure](#) function for the specified output channel. This function compares the array in **reportedOutputs** to the array in **measuredOutputs** and calculates the calibration constants for the voltage measurements returned by the `niDCPower_Measure` function. Refer to the calibration procedure for the device you are calibrating for detailed instructions on the appropriate use of this function. This function can only be called in an external calibration session.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower_InitExtCal function.
channelName	ViConstString	Specifies the channel name to which these calibration settings apply.
range	ViReal64	Specifies the range to calibrate with these settings. Only one channel at a time may be calibrated.
numberOfMeasurements	ViUInt32	Specifies the number of elements in reportedOutputs and measuredOutputs .
reportedOutputs	ViReal64[]	Specifies an array of the output values that were returned by the niDCPower_Measure function.
measuredOutputs	ViReal64[]	Specifies an array of the output values measured by an external precision digital multimeter.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_CalAdjustCurrentLimit

ViStatus niDCPower_CalAdjustCurrentLimit (ViSession vi, ViConstString
channelName, ViReal64 range, ViUInt32 numberOfMeasurements,
ViReal64 requestedOutputs[], ViReal64 measuredOutputs[]);

Purpose

Calculates the calibration constants for the current limit for the specified output channel and range. This function compares the array in **requestedOutputs** to the array in **measuredOutputs** and calculates the calibration constants for the current limit returned by the device. Refer to the calibration procedure for the power supply you are calibrating for detailed instructions on the appropriate use of this function. This function can only be called from an external calibration session.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
channelName	ViConstString	Specifies the channel name to which these calibration settings apply.
range	ViReal64	Specifies the range to calibrate with these settings. Only one channel at a time may be calibrated.
numberOfMeasurements	ViUInt32	Specifies the number of elements in requestedOutputs and measuredOutputs .
requestedOutputs	ViReal64[]	Specifies an array of the output values that were requested in the niDCPower_ConfigureCurrentLimit function.
measuredOutputs	ViReal64[]	Specifies an array of the output values measured by an external precision digital multimeter.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_CalAdjustCurrentMeasurement

```
ViStatus niDCPower_CalAdjustCurrentMeasurement (ViSession vi,  
        ViConstString channelName, ViReal64 range, ViUInt32  
        numberOfMeasurements, ViReal64 reportedOutputs[], ViReal64  
        measuredOutputs[]);
```

Purpose

Calibrates the current measurements returned by the [niDCPower_Measure](#) function for the specified output channel. This function calculates the calibration constants for the array in **reportedOutputs** to the array in **measuredOutputs** and calculates the calibration constants for the current measurements returned by the `niDCPower_Measure` function. Refer to the calibration procedure for the device you are calibrating for detailed instructions about the appropriate use of this function. This function can only be called in an external calibration session.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower_InitExtCal function.
channelName	ViConstString	Specifies the output channel name to which these calibration settings apply.
range	ViReal64	Specifies the range to calibrate with these settings. Only one channel at a time may be calibrated.
numberOfMeasurements	ViUInt32	Specifies the number of elements in reportedOutputs and measuredOutputs .
reportedOutputs	ViReal64[]	Specifies an array of the output values that were returned by the niDCPower_Measure function.
measuredOutputs	ViReal64[]	Specifies an array of the output values measured by an external precision digital multimeter.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ChangeExtCalPassword

ViStatus niDCPower_ChangeExtCalPassword (ViSession vi, ViConstString
oldPassword, ViConstString newPassword);

Purpose

Changes the **password** that is required to initialize an external calibration session. The **password** can be a maximum of four alphanumeric characters. If you call this function in a regular session, **password** is changed immediately. If you call this function in an external calibration session, **password** is changed only after you close the session using the [niDCPower_CloseExtCal](#) function with **action** set to NIDCPOWER_VAL_COMMIT.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower InitExtCal function.
oldPassword	ViConstString	Specifies the previous password used to protect the calibration values.
newPassword	ViConstString	Specifies the new password to use to protect the calibration values.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetExtCalRecommendedInterval

ViStatus niDCPower_GetExtCalRecommendedInterval (ViSession vi, ViInt32
*months);

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower_InitExtCal function.

Output

Name	Type	Description
months	ViInt32*	Specifies the recommended maximum interval, in months , between external calibrations in months.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetExtCalLastDateAndTime

ViStatus niDCPower_GetExtCalLastDateAndTime (ViSession vi, ViInt32 *year,
ViInt32 *month, ViInt32 *day, ViInt32 *hour, ViInt32 *minute);

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower_InitExtCal function.

Output

Name	Type	Description
year	ViInt32*	Returns the year the device was last calibrated.
month	ViInt32*	Returns the month in which the device was last calibrated.
day	ViInt32*	Returns the day on which the device was last calibrated.
hour	ViInt32*	Returns the hour (in 24-hour time) in which the device was last calibrated.
minute	ViInt32*	Returns the minute in which the device was last calibrated.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetCalUserDefinedInfoMaxSize

```
ViStatus niDCPower_GetCalUserDefinedInfoMaxSize (ViSession vi,  
ViInt32*infoSize);
```

Purpose

Returns the maximum number of characters that can be used to store user-defined information in the device onboard EEPROM.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or the niDCPower_InitWithOptions function.

Output

Name	Type	Description
infoSize	ViInt32*	Returns the number of characters that can be stored in the device onboard EEPROM.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_SetCalUserDefinedInfo

```
ViStatus niDCPower_SetCalUserDefinedInfo (ViSession vi, ViConstString  
info);
```

Purpose

Stores a user-defined string of characters in the device onboard EEPROM. If the string is longer than the maximum allowable size, it is truncated. This function overwrites any existing user-defined information. If you call this function in a regular session, **info** is immediately changed. If you call this function in an external calibration session, **info** is changed only after you close the session using the [niDCPower_CloseExtCal](#) function with **action** set to NIDCPOWER_VAL_COMMIT.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
info	ViConstString	Specifies the string to store in the device onboard EEPROM.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetCalUserDefinedInfo

ViStatus niDCPower_GetCalUserDefinedInfo (ViSession vi, ViString info);

Purpose

Returns the user-defined information in the device onboard EEPROM.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower InitExtCal function.

Output

Name	Type	Description
info	ViString	Returns the user-defined information stored in the device onboard EEPROM.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ReadCurrentTemperature

ViStatus niDCPower_ReadCurrentTemperature (ViSession vi, ViReal64
*temperature);

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Output

Name	Type	Description
temperature	ViReal64*	Returns the onboard temperature , in degrees Celsius, of the device.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetExtCalLastTemp

```
ViStatus niDCPower_GetExtCalLastTemp (ViSession vi, ViReal64  
    *temperature);
```

Purpose

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

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument calibration session. vi is obtained from the niDCPower InitExtCal function.

Output

Name	Type	Description
temperature	ViReal64*	Returns the onboard temperature of the device, in degrees Celsius, during the last successful external calibration.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_InitWithOptions

```
ViStatus niDCPower_InitWithOptions (ViRsrc resourceName, ViBoolean  
    IDQuery, ViBoolean resetDevice, ViString optionString, ViSession  
    *newVi);
```

Purpose

Creates a new IVI instrument driver session to the device specified in **resourceName** and returns a session handle you use to identify the device in all subsequent NI-DCPower function calls. With this function, you can optionally set the initial state of the following session attributes:

- [NIDCPOWER_ATTR_SIMULATE](#)
- [NIDCPOWER_ATTR_DRIVER_SETUP](#)
- [NIDCPOWER_ATTR_RANGE_CHECK](#)
- [NIDCPOWER_ATTR_QUERY_INSTRUMENT_STATE](#)
- [NIDCPOWER_ATTR_CACHE](#)
- [NIDCPOWER_ATTR_RECORD_COERCIONS](#)

This function also sends initialization commands to set the device to the state necessary for NI-DCPower to operate.

To place the device in a known start-up state when creating a new session, set **resetDevice** to VI_TRUE. This action is equivalent to using the [niDCPower_reset](#) function.

To open a session and leave the device in its existing configuration without passing through a transitional output state, set **resetDevice** to VI_FALSE, and immediately call the [niDCPower_Abort](#) function. To apply a new configuration without disrupting the output channels of the device, configure the device in [Delayed Configuration](#) mode as in the previous session changing only the desired settings, and then call the [niDCPower_Initiate](#) function.

Parameters

Input

Name	Type	Description
resourceName	ViRsrc	Specifies the resourceName assigned by Measurement & Automation Explorer (MAX), for example "PXI1Slot3" where "PXI1Slot3" is an instrument's resourceName . resourceName can also be a logical IVI name.
IDQuery	ViBoolean	Specifies whether the device is queried to determine if the device is a valid instrument for NI-DCPower. The default value is VI_TRUE.
resetDevice	ViBoolean	Specifies whether to reset the device during the initialization procedure. The default value is VI_TRUE.
optionString	ViString	<p>Specifies the initial value of certain attributes for the session. The syntax for optionString is a list of attributes with an assigned value where 1 is VI_TRUE and 0 is VI_FALSE. Each attribute/value combination is delimited with a comma, as shown in the following example:</p> <pre>"Simulate=0,RangeCheck=1,QueryInstrStatus=0,Cache=1"</pre> <p>If you do not wire this input or pass an empty string, the session assigns the default values, shown in the example, for these attributes. You do not have to specify a value for all the attributes. If you do not specify a value for an attribute, the default value is used.</p> <p>For more information about simulating a device, refer to Simulating a Power Supply or SMU.</p>

Output

Name	Type	Description
newVi	ViSession*	Returns a handle that you use to identify the device in all subsequent NI-DCPower function calls.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_Disable

ViStatus niDCPower_Disable (ViSession vi);

Purpose

Places the device in a quiescent state where it has minimal or no impact on the system to which it is connected. The power output and all exported signals are disabled. This function performs the same actions as the [niDCPower_reset](#) function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ResetDevice

ViStatus niDCPower_ResetDevice (ViSession vi);

Purpose

Resets the device to a known state. The function disables power generation, resets session attributes to their default values, clears errors such as overtemperature and unexpected loss of auxiliary power, commits the session attributes, and leaves the session in [Immediate mode](#).

The `niDCPower_ResetDevice` function performs a hard reset on the device and the driver software. This function has the same functionality as using reset in Measurement & Automation Explorer.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ResetWithDefaults

ViStatus niDCPower_ResetWithDefaults (ViSession vi);

Purpose

Resets the device to a known state. This function disables power generation, resets session attributes to their default values, clears errors such as overtemperature and unexpected loss of auxiliary power, commits the session attributes, and leaves the session in [Immediate mode](#). In addition to exhibiting the behavior of the [niDCPower_reset](#) function, this function can assign user-defined default values for configurable attributes from the IVI configuration.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetChannelName

```
ViStatus niDCPower_GetChannelName (ViSession vi, ViInt32 index, ViInt32  
    bufferSize, ViChar channelName[]);
```

Purpose

Retrieves the output **channelName** that corresponds to the requested **index**. Use the [NIDCPOWER_ATTR_CHANNEL_COUNT](#) attribute to determine the upper bound of valid values for **index**.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
index	ViInt32	Specifies which output channel name to return. The index values begin at 1.
bufferSize	ViInt32	<p>Specifies the number of bytes in the ViChar array you specify for channelName. If the channelName, including the terminating NUL byte, contains more bytes than you indicate in this attribute, the function copies (buffer size - 1) bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is 123456 and the buffer size is 4, the function places 123 into the buffer and returns 7.</p> <p>If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.</p> <p>If you pass 0, you can pass VI_NULL for channelName.</p>

Output

Name	Type	Description
channelName	ViChar[]	Returns the output channel name that corresponds to index .

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call niDCPower_error_message . To obtain additional information concerning the error condition, call niDCPower_GetError .
--------	----------	--

niDCPower_GetNextCoercionRecord

```
ViStatus niDCPower_GetNextCoercionRecord (ViSession vi, ViInt32  
    bufferSize, ViChar coercionRecord[]);
```

Purpose

Returns the coercion information associated with the IVI session and clears the earliest instance in which NI-DCPower coerced a value you specified.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
bufferSize	ViInt32	Specifies the number of bytes in the ViChar array you specify for coercionRecord . If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate in this attribute, the function copies (buffer size - 1) bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is 123456 and the buffer size is 4, the function places 123 into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI_NULL for **coercionRecord**.

Output

Name	Type	Description
coercionRecord	ViChar[]	Returns the next coercionRecord for the IVI session. If there are no coercionRecords , the function returns an empty string.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call niDCPower_error_message . To obtain additional information concerning the error condition, call niDCPower_GetError .
---------------	----------	--

niDCPower_ClearInterchangeWarnings

ViStatus niDCPower_ClearInterchangeWarnings (ViSession vi);

Purpose

Clears the list of current interchange warnings.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_ResetInterchangeCheck

ViStatus niDCPower_ResetInterchangeCheck (ViSession vi);

Purpose

When developing a complex test system that consists of multiple test modules, it is generally a good idea to design the test modules so that they can run in any order. To do so requires ensuring that each test module completely configures the state of each instrument it uses. If a particular test module does not completely configure the state of an instrument, the state of the instrument depends on the configuration from a previously executed test module. If you execute the test modules in a different order, the behavior of the instrument and therefore the entire test module is likely to change. This change in behavior is generally instrument specific and represents an interchangeability problem.

You can use this function to test for such cases. After you call this function, the interchangeability checking algorithms in the specific driver ignore all previous configuration operations. By calling this function at the beginning of a test module, you can determine whether the test module has dependencies on the operation of previously executed test modules.

This function does not clear the interchangeability warnings from the list of previously recorded interchangeability warnings. If you want to guarantee that the [niDCPower_GetNextInterchangeWarning](#) function only returns those interchangeability warnings that are generated after calling this function, you must clear the list of interchangeability warnings. You can clear the interchangeability warnings list by repeatedly calling the [niDCPower_GetNextInterchangeWarning](#) function until no more interchangeability warnings are returned. If you are not interested in the content of those warnings, you can call the [niDCPower_ClearInterchangeWarnings](#) function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_GetNextInterchangeWarning

```
ViStatus niDCPower_GetNextInterchangeWarning (ViSession vi, ViInt32  
    bufferSize, ViChar interchangeWarning[]);
```

Purpose

This function returns the interchangeability warning associated with the IVI session. It retrieves and clears the earliest instance in which the class driver recorded an interchangeability warning. Interchangeability warnings indicate that using your application with a different device may cause a different behavior.

NI-DCPower performs interchangeability checking when the [NIDCPOWER_ATTR_INTERCHANGE_CHECK](#) attribute is set to VI_TRUE. This function returns an empty string in warning if no interchangeability warnings remain for the session. In general, NI-DCPower generates interchangeability warnings when an attribute that affects the behavior of the device is in a state that you did not specify.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
bufferSize	ViInt32	Specifies the number of bytes in the ViChar array you specify for interchangeWarning . If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate in this attribute, the function copies (buffer size - 1) bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is 123456 and the buffer size is 4, the function places 123 into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value. If you pass 0, you can pass VI_NULL for interchangeWarning .

Output

Name	Type	Description
interchangeWarning	ViChar[]	Returns the next interchange warning for the IVI session. If there are no interchange warnings, the function returns an empty string.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call niDCPower_error_message . To obtain additional information concerning the error condition, call niDCPower_GetError .
---------------	----------	--

niDCPower_GetError

ViStatus niDCPower_GetError(ViSession vi, ViStatus *Code, ViInt32
bufferSize, ViChar description[])

Purpose

Retrieves and then clears the IVI error information for the session or the current execution thread unless **bufferSize** is 0, in which case the function does not clear the error information. By passing 0 for the buffer size, you can ascertain the buffer size required to get the entire error description string and then call the function again with a sufficiently large buffer size.

If the user specifies a valid IVI session for **vi**, this function retrieves and then clears the error information for the session. If the user passes `VI_NULL` for **vi**, this function retrieves and then clears the error information for the current execution thread. If **vi** is an invalid session, the function does nothing and returns an error. Normally, the error information describes the first error that occurred since the user last called `niDCPower_GetError` or [niDCPower_ClearError](#).

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.
bufferSize	ViInt32	Specifies the number of bytes in the ViChar array you specify for description . If the error description, including the terminating NUL byte, contains more bytes than you indicate in this attribute, the function copies (buffer size - 1) bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is 123456 and the buffer size is 4, the function places 123 into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value. If you pass 0 for this attribute, you can pass VI_NULL for description .

Output

Name	Type	Description
Code	ViStatus*	Returns the error code for the session or execution thread.
description	ViChar[]	Returns the error description for the IVI session or execution thread. If there is no description, the function returns an empty string. The buffer must contain at least as many elements as the value you specify with bufferSize . If the error description, including the terminating NUL byte, contains more bytes than you indicate with bufferSize , the function copies (buffer size - 1) bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is 123456 and the buffer size is 4, the function places 123 into the buffer and returns 7. If you pass 0 for bufferSize , you can pass VI_NULL for this attribute.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call niDCPower_error_message . To obtain additional information concerning the error condition, call niDCPower_GetError.
--------	----------	---

niDCPower_ClearError

ViStatus niDCPower_ClearError(ViSession vi)

Purpose

Clears the error code and error description for the IVI session. If the user specifies a valid IVI session for **vi**, this function clears the error information for the session. If the user passes VI_NULL for **vi**, this function clears the error information for the current execution thread. If the ViSession parameter is an invalid session, the function does nothing and returns an error.

The function clears the error code by setting it to VI_SUCCESS. If the error description string is non-NULL, the function de-allocates the error description string and sets the address to VI_NULL.

Maintaining the error information separately for each thread is useful if the user does not have a session handle to pass to the [niDCPower_GetError](#) function, which occurs when a call to [niDCPower_init](#) or [niDCPower_InitWithOptions](#) fails.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_LockSession

ViStatus niDCPower_LockSession(ViSession vi, ViBoolean *callerHasLock)

Purpose

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

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

- The application called the `niDCPower_LockSession` function.
- A call to NI-DCPower locked the session.
- A call to the IVI engine locked the session.
- After a call to the `niDCPower_LockSession` function returns successfully, no other threads can access the device session until you call the [niDCPower_UnlockSession](#) function.
- Use the `niDCPower_LockSession` function and the `niDCPower_UnockSession` function around a sequence of calls to instrument driver functions if you require that the device retain its settings through the end of the sequence.

You can safely make nested calls to the `niDCPower_LockSession` function within the same thread. To completely unlock the session, you must balance each call to the `niDCPower_LockSession` function with a call to the `niDCPower_UnockSession` function. If, however, you use **Caller_Has_Lock** in all calls to the `niDCPower_LockSession` and `niDCPower_UnockSession` function within a function, the IVI Library locks the session only once within the function regardless of the number of calls you make to the `niDCPower_LockSession` function. This behavior allows you to call the `niDCPower_UnockSession` function just once at the end of the function.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Output

Name	Type	Description
callerHasLock	ViBoolean*	This parameter is optional. If you do not want to use this parameter, pass VI_NULL.

Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to VI_FALSE. Pass the address of the same local variable to any other calls you make to the niDCPower_LockSession function or the niDCPower_UnlockSession function in the same function.

The parameter is an input/output parameter. The niDCPower_LockSession and niDCPower_UnlockSession functions each inspect the current value and take the following actions.

- If the value is VI_TRUE, the niDCPower_LockSession function does not lock the session again.
- If the value is VI_FALSE, the niDCPower_LockSession function obtains the lock and sets the value of the parameter to VI_TRUE.
- If the value is VI_FALSE, the niDCPower_UnlockSession function does not attempt to unlock the session.
- If the value is VI_TRUE, the niDCPower_UnlockSession function releases the lock and sets the value of the parameter to VI_FALSE.

Thus, you can, call the niDCPower_UnlockSession function at the end of your function without worrying about whether you actually have the lock, as shown in the following example.

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
{
    ViStatus error = VI_SUCCESS;
    ViBoolean haveLock = VI_FALSE;

    if (flags & BIT_1)
    {
        viCheckErr( niDCPower_LockSession(vi, &haveLock));
        viCheckErr( TakeAction1(vi));
    }
    if (flags & BIT_2)
    {
        viCheckErr( niDCPower_UnlockSession(vi, &haveLock));
        viCheckErr( TakeAction2(vi));
    }
    viCheckErr( niDCPower_LockSession(vi, &haveLock));
}
if (flags & BIT_3)
```

```
viCheckErr( TakeAction3(vi));  
}
```

Error:

```
/*At this point, you cannot really be sure that you have the lock. Fortunately, the  
haveLock variable takes care of that for you.*/  
niDCPower_UnlockSession(vi, &haveLock);  
return error;  
}
```

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

niDCPower_UnlockSession

ViStatus niDCPower_UnlockSession(ViSession vi, ViBoolean *callerHasLock)

Purpose

Releases a lock that you acquired on an device session using [niDCPower_LockSession](#). Refer to niDCPower_LockSession for additional information on session locks.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Output

Name	Type	Description
callerHasLock	ViBoolean*	this attribute is optional. If you do not want to use this attribute, pass VI_NULL.

Use this attribute in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session.

Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to VI_FALSE. Pass the address of the same local variable to any other calls you make to niDCPower_LockSession or niDCPower_UnlockSession in the same function.

The parameter is an input/output parameter. niDCPower_LockSession and niDCPower_UnlockSession each inspect the current value and take the following actions.

- If the value is VI_TRUE, niDCPower_LockSession does not lock the session again.
- If the value is VI_FALSE, niDCPower_LockSession obtains the lock and sets the value of the parameter to VI_TRUE.
- If the value is VI_FALSE, niDCPower_UnlockSession does not attempt to unlock the session.
- If the value is VI_TRUE, niDCPower_UnlockSession releases the lock and sets the value of the parameter to VI_FALSE.

Thus, you can, call niDCPower_UnlockSession at the end of your function without worrying about whether you actually have the lock, as the following example shows.

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
{
    ViStatus error = VI_SUCCESS;
    ViBoolean haveLock = VI_FALSE;

    if (flags & BIT_1)
    {
        viCheckErr( niDCPower_LockSession(vi, &haveLock));
        viCheckErr( TakeAction1(vi));
    }
    if (flags & BIT_2)
    {
        viCheckErr( niDCPower_UnlockSession(vi, &haveLock));
        viCheckErr( TakeAction2(vi));
    }
    viCheckErr( niDCPower_LockSession(vi, &haveLock));
}
if (flags & BIT_3)
```

```
viCheckErr( TakeAction3(vi));  
}
```

Error:

```
/*At this point, you cannot really be sure that you have the lock. Fortunately, the  
haveLock variable takes care of that for you.*/  
niDCPower_UnlockSession(vi, &haveLock);  
return error;  
}
```

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

NIDCPOWER_ATTR_OUTPUT_ENABLED

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Channel	None	niDCPower_ConfigureOutputEnabled

Description

Specifies whether the output is enabled or disabled.

Depending on the set value for the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute, the [NIDCPOWER_ATTR_VOLTAGE_LEVEL](#) attribute or the [NIDCPOWER_ATTR_CURRENT_LEVEL](#) attribute must be set in addition to enabling the output.



Note If the device is in [Delayed Configuration mode](#), enabling the output will not take effect until you call the [niDCPower_Initiate](#) function.

Defined Values:

VI_TRUE	Enables generation on the specified channels.
VI_FALSE	Disables generation on the specified channels.

Default Value: VI_FALSE

NIDCPOWER_ATTR_OUTPUT_FUNCTION

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	niDCPower_ConfigureOutputFunction

Description

Configures the function to generate on the specified channel(s).

When `NIDCPOWER_VAL_DC_VOLTAGE` is selected, the device generates the desired voltage level on the output as long as the output current is below the current limit. The following attributes can be used to configure the channel when `NIDCPOWER_VAL_DC_VOLTAGE` is selected:

- [NIDCPOWER_ATTR_VOLTAGE_LEVEL](#)
- [NIDCPOWER_ATTR_CURRENT_LIMIT](#)
- [NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE](#)
- [NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE](#)

When `NIDCPOWER_VAL_DC_CURRENT` is selected, the device generates the desired current level on the output as long as the output voltage is below the voltage limit. The following attributes can be used to configure the channel when `NIDCPOWER_VAL_DC_CURRENT` is selected:

- [NIDCPOWER_ATTR_CURRENT_LEVEL](#)
- [NIDCPOWER_ATTR_VOLTAGE_LIMIT](#)
- [NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE](#)
- [NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE](#)

Defined Values:

<code>NIDCPOWER_VAL_DC_VOLTAGE</code>	Sets the output function to DC voltage.
<code>NIDCPOWER_VAL_DC_CURRENT</code>	Sets the output function to DC current.

Default Value: `NIDCPOWER_VAL_DC_VOLTAGE`

NIDCPOWER_ATTR_SENSE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	niDCPower_ConfigureSense

Description

Selects either [local](#) or [remote](#) sensing of the output voltage for the specified channel(s). Refer to the *Devices* topic specific to your device in the *NI DC Power Supplies and SMUs Help* to find out more information about sensing on supported channels.

Defined Values:

NIDCPOWER_VAL_LOCAL	Local sensing is selected.
NIDCPOWER_VAL_REMOTE	Remote sensing is selected.

Default Value: NIDCPOWER_VAL_LOCAL

NIDCPOWER_ATTR_VOLTAGE_LEVEL

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	None	niDCPower_ConfigureVoltageLevel

Description

Specifies the voltage level, in volts, the device attempts to generate on the specified channel(s).

The `NIDCPOWER_ATTR_VOLTAGE_LEVEL` attribute is applicable only if the `NIDCPOWER_ATTR_OUTPUT_FUNCTION` attribute is set to `NIDCPOWER_VAL_DC_VOLTAGE`.

The channel must be enabled for the specified voltage level to take effect. Refer to the `NIDCPOWER_ATTR_OUTPUT_ENABLED` attribute for more information about enabling the output channel.

Valid Values:

The valid values for this attribute are defined by the values to which the `NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE` attribute is set.

NIDCPOWER_ATTR_CURRENT_LIMIT

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	None	niDCPower_ConfigureCurrentLimit

Description

Specifies the current limit, in amps, for the output not to exceed when generating the desired voltage level on the specified channels.

The `NIDCPOWER_ATTR_CURRENT_LIMIT` attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to `NIDCPOWER_VAL_DC_VOLTAGE`.

The channel must be enabled for the specified current limit to take effect. Refer to the [NIDCPOWER_ATTR_OUTPUT_ENABLED](#) attribute for more information about enabling the output channel.

Valid Values:

The valid values for this attribute are defined by the values to which [NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE](#) attribute is set.

NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	Yes	niDCPower_ConfigureVoltageLevelRange

Description

Specifies the voltage level range, in volts, for the specified channel(s). The range defines the valid values to which the voltage level can be set. Use the [NIDCPOWER_ATTR_VOLTAGE_LEVEL_AUTORANGE](#) attribute to enable automatic selection of the voltage level range.

The `NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE` attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to `NIDCPOWER_VAL_DC_VOLTAGE`.

The channel must be enabled for the specified voltage level range to take effect. Refer to the [NIDCPOWER_ATTR_OUTPUT_ENABLED](#) attribute for more information about enabling the output channel.

Valid Values:

NI PXI-4110

Channel	Voltage Level Range (V)	Voltage Level (V)
0	6	0 to +6
1	20	0 to +20
2	20	0 to -20

NI PXI-4130

Channel	Voltage Level Range (V)	Voltage Level (V)
0	6	0 to +6
1	6	-6 to +6
	20	-20 to +20



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest level. For example, requesting the 10 V voltage level range on Channel 1 on the NI-PXI 4130 coerces the voltage level range to 20 V. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	N/A	Yes	niDCPower_ConfigureCurrentLimitRange

Description

Specifies the current limit range, in amps, for the specified channel(s). The range defines the valid value to which the current limit can be set. Use the [NIDCPOWER_ATTR_CURRENT_LIMIT_AUTORANGE](#) attribute to enable automatic selection of the current limit range.

The `NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE` attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to `NIDCPOWER_VAL_DC_VOLTAGE`.

The channel must be enabled for the specified current limit to take effect. Refer to the [NIDCPOWER_ATTR_OUTPUT_ENABLED](#) attribute for more information about enabling the output channel.

Valid Values:

NI PXI-4110

Channel	Current Limit Range	Current Limit
0	1 A	+0.01 to +1 A
1, 2	20 mA	+0.20 to +20 mA
	1 A	+0.01 to +1 A

NI PXI-4130

Channel	Current Limit Range	Current Limit
0	1 A	+0.02 to +1 A
1	200 μ A	+4 to +200 μ A
	2 mA	+0.04 to +2 mA
	20 mA	+0.40 to +20 mA
	200 mA	+4 to +200 mA
	2 A	+0.04 to +2 A



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest range. For example, requesting the 100 mA current limit range on Channel 1 on the NI-PXI 4130 coerces the current level range to 200 mA. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

NIDCPOWER_ATTR_VOLTAGE_LEVEL_AUTORA

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	None

Description

Specifies whether or not NI-DCPower automatically selects the voltage level range based on the desired voltage level for the specified channel(s).

If this attribute is set to `NIDCPOWER_VAL_ON`, NI-DCPower ignores any changes you make to the [NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE](#) attribute. If you change the `NIDCPOWER_ATTR_VOLTAGE_LEVEL_AUTORANGE` attribute from `NIDCPOWER_VAL_ON` to `NIDCPOWER_VAL_OFF`, NI-DCPower remembers the last value the `NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE` attribute was set to (or the default value if the attribute was never set) and uses that value as the voltage level range.

Query the `NIDCPOWER_ATTR_VOLTAGE_LEVEL_RANGE` attribute by using the [niDCPower_GetAttributeViInt32](#) function to find out which range NI-DCPower automatically selects.

The `NIDCPOWER_ATTR_VOLTAGE_LEVEL_AUTORANGE` attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to `NIDCPOWER_VAL_DC_VOLTAGE`.

Defined Values:

<code>NIDCPOWER_VAL_ON</code>	NI-DCPower automatically selects the voltage level range.
<code>NIDCPOWER_VAL_OFF</code>	NI-DCPower does not automatically select the voltage level range.

Default Value: `NIDCPOWER_VAL_OFF`

NIDCPOWER_ATTR_CURRENT_LIMIT_AUTORAN

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	None

Description

Specifies whether or not NI-DCPower automatically selects the current limit range based on the desired current limit for the specified channels.

If this attribute is set to NIDCPOWER_VAL_ON, NI-DCPower ignores any changes you make to the [NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE](#) attribute. If you change this attribute from NIDCPOWER_VAL_ON to NIDCPOWER_VAL_OFF, NI-DCPower remembers the last value the NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE attribute was set to (or the default value if the attribute was never set) and uses that value as the current limit range.

Query the NIDCPOWER_ATTR_CURRENT_LIMIT_RANGE attribute by using the [niDCPower_GetAttributeViInt32](#) function to find out which range NI-DCPower automatically selects.

The NIDCPOWER_ATTR_CURRENT_LIMIT_AUTORANGE attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to NIDCPOWER_VAL_DC_VOLTAGE.

Defined Values:

NIDCPOWER_VAL_ON	NI-DCPower automatically selects the current limit range.
NIDCPOWER_VAL_OFF	NI-DCPower does not automatically select the current limit range.

Default Value: NIDCPOWER_VAL_OFF

NIDCPOWER_ATTR_CURRENT_LEVEL

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	None	niDCPower_ConfigureCurrentLevel

Description

Specifies the current level, in amps, the device attempts to generate on the specified channel(s).

This attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to NIDCPOWER_VAL_DC_CURRENT.

The channel must be enabled for the specified current level to take effect. Refer to the [NIDCPOWER_ATTR_OUTPUT_ENABLED](#) attribute for more information about enabling the output channel.

Valid Values:

The valid values for this attribute are defined by the values to which the [NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE](#) attribute is set.

NIDCPOWER_ATTR_VOLTAGE_LIMIT

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	None	niDCPower_ConfigureVoltageLimit

Description

Specifies the voltage limit, in volts, for the output not to exceed when generating the desired current level on the specified channels.

This attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to NIDCPOWER_VAL_DC_CURRENT.

The channel must be enabled for the specified current level to take effect. Refer to the [NIDCPOWER_ATTR_OUTPUT_ENABLED](#) attribute for more information about enabling the output channel.

Valid Values:

The valid values for this attribute are defined by the values to which the [NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE](#) attribute is set.

NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	Yes	niDCPower_ConfigureCurrentLevelRange

Description

Specifies the current level range, in amps, for the specified channel(s). The range defines the valid value to which the current level can be set. Use the [NIDCPOWER_ATTR_CURRENT_LEVEL_AUTORANGE](#) attribute to enable automatic selection of the current level range.

The `NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE` attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to `NIDCPOWER_VAL_DC_CURRENT`.

The channel must be enabled for the specified current level range to take effect. Refer to the [NIDCPOWER_ATTR_OUTPUT_ENABLED](#) attribute for more information about enabling the output channel.

Valid Values:

NI PXI-4110

Channel	Current Level Range	Current Level
0	1 A	+0.01 to +1 A
1	20 mA	+0.20 to +20 mA
	1 A	+0.01 to +1 A
2	20 mA	-0.20 to -20 mA
	1 A	-0.01 to -1 A

NI PXI-4130

Channel	Current Level Range	Current Level
0	1 A	+0.02 to +1 A
1	200 μ A	+4 to +200 μ A and -4 to -200 μ A
	2 mA	+0.04 to +2 mA and -0.04 to -2 mA
	20 mA	+0.40 to +20 mA and -0.40 to -20 mA
	200 mA	+4 to +200 mA and -4 to -200 mA
	2 A	+0.04 to +2 A and -0.04 to -2 A



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest range. For example, requesting the 100 mA current level range on Channel 1 on the NI-PXI 4130 coerces the current level to the 200 mA current level range. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	Yes	niDCPower_ConfigureVoltageLimitRange

Description

Specifies the voltage limit range, in volts, for the specified channel(s). The range defines the valid values to which the voltage limit can be set. Use the [NIDCPOWER_ATTR_VOLTAGE_LIMIT_AUTORANGE](#) attribute to enable automatic selection of the voltage limit range.

The `NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE` attribute is applicable only if the channel is set to `NIDCPOWER_VAL_DC_CURRENT` in the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute.

The channel must be enabled for the specified voltage limit range to take effect. Refer to the [NIDCPOWER_ATTR_OUTPUT_ENABLED](#) attribute for more information about enabling the output channel.

Valid Values:

NI PXI-4110

Channel	Voltage Limit Range (V)	Voltage Limit (V)
0	6	0 to +6
1, 2	20	0 to +20

NI PXI-4130

Channel	Voltage Limit Range (V)	Voltage Limit (V)
0	6	0 to +6
1	6	0 to +6
	20	0 to +20



Note If a range other than what is listed in the preceding table is selected, it will be coerced to the next-highest range. For example, requesting the 10 V voltage limit range on Channel 1 on the NI-PXI 4130 coerces the voltage limit range to 20 V. Refer to the [Ranges](#) topic in the *NI DC Power Supplies and SMUs Help* for more information about coercion.

NIDCPOWER_ATTR_CURRENT_LEVEL_AUTORA

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	None

Description

Specifies whether or not NI-DCPower automatically selects the current level range based on the desired current level for the specified channels.

If this attribute is set to NIDCPOWER_VAL_ON, NI-DCPower ignores any changes you make to the

[NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE](#) attribute. If you change the NIDCPOWER_ATTR_CURRENT_LEVEL_AUTORANGE attribute from NIDCPOWER_VAL_ON to NIDCPOWER_VAL_OFF, NI-DCPower remembers the last value the

NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE attribute was set to (or the default value if the attribute was never set) and uses that value as the current level range.

Query the NIDCPOWER_ATTR_CURRENT_LEVEL_RANGE attribute by using the [niDCPower_GetAttributeViInt32](#) function to find out which range NI-DCPower automatically selects.

The NIDCPOWER_ATTR_CURRENT_LEVEL_AUTORANGE attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to NIDCPOWER_VAL_DC_CURRENT.

Defined Values:

NIDCPOWER_VAL_ON	NI-DCPower automatically selects the current level range.
NIDCPOWER_VAL_OFF	NI-DCPower does not automatically select the current level range.

Default Value: NIDCPOWER_VAL_OFF

NIDCPOWER_ATTR_VOLTAGE_LIMIT_AUTORAN

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	None

Description

Specifies whether or not NI-DCPower automatically selects the voltage limit range based on the desired voltage limit for the specified channel(s).

If this attribute is set to NIDCPOWER_VAL_ON, NI-DCPower ignores any changes you make to the

[NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE](#) attribute. If you change the NIDCPOWER_ATTR_VOLTAGE_LIMIT_AUTORANGE attribute from NIDCPOWER_VAL_ON to NIDCPOWER_VAL_OFF, NI-DCPower remembers the last value the NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE attribute was set to (or the default value if the attribute was never set) and uses that value as the voltage limit range.

Query the NIDCPOWER_ATTR_VOLTAGE_LIMIT_RANGE attribute by using the [niDCPower_GetAttributeViInt32](#) function to find out which range NI-DCPower automatically selects.

The NIDCPOWER_ATTR_VOLTAGE_LIMIT_AUTORANGE attribute is applicable only if the [NIDCPOWER_ATTR_OUTPUT_FUNCTION](#) attribute is set to NIDCPOWER_VAL_DC_CURRENT.

Defined Values:

NIDCPOWER_VAL_ON	NI-DCPower automatically selects the voltage limit range.
NIDCPOWER_VAL_OFF	NI-DCPower does not automatically select the voltage limit range.

Default Value: NIDCPOWER_VAL_OFF

NIDCPOWER_ATTR_POWER_SOURCE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Device	None	None

Description

Specifies the power source to use. NI-DCPower switches the power source used by the device to the specified value.

Defined Values:

NIDCPOWER_VAL_INTERNAL	Uses the PXI chassis power source.
NIDCPOWER_VAL_AUXILIARY	Uses the auxiliary power source connected to the device.
NIDCPOWER_VAL_AUTOMATIC	Uses the auxiliary power source if it is available; otherwise uses the PXI chassis power source.

Default Value: NIDCPOWER_VAL_AUTOMATIC



Note Automatic selection is not persistent and occurs only at the time this attribute is set to NIDCPOWER_VAL_AUTOMATIC. However, if the session was in [Delayed Configuration mode](#) when you set this attribute, the power source selection only occurs after you call the [niDCPower_Initiate](#) function.

NIDCPOWER_ATTR_POWER_SOURCE_IN_USE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	RO	Device	None	None

Description

Indicates whether the device is using the internal or auxiliary power source to generate power.

Defined Values:

NIDCPOWER_VAL_INTERNAL	Uses the PXI chassis power source.
NIDCPOWER_VAL_AUXILIARY	Uses the auxiliary power source connected to the device.

NIDCPOWER_ATTR_AUXILIARY_POWER_SOURC

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	RO	Device	None	None

Description

Indicates whether an auxiliary power source is connected to the device. A value of VI_FALSE may indicate the auxiliary input fuse has blown. Refer to the [Detecting Internal/Auxiliary Power](#) topic for more information about auxiliary power.



Note This attribute does not necessarily indicate if the device is using the auxiliary power source to generate power. Use the [NIDCPOWER ATTR POWER SOURCE IN USE](#) attribute to retrieve this information.

NIDCPOWER_ATTR_SAMPLES_TO_AVERAGE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	None

Description

Specifies the number of samples to average when you take a measurement. Increasing the number of samples to average decreases measurement noise but increases the time required to take a measurement. Refer to the *Measurement Averaging* topic specific to your device in the *NI DC Power Supplies and SMUs Help* for optional attribute settings to improve immunity to certain types of noise.

When you set the `NIDCPOWER_ATTR_SAMPLES_TO_AVERAGE` attribute in [Immediate mode](#), the output channel measurements might move out of synchronization. While NI-DCPower automatically synchronizes measurements upon the initialization of a session, you can force a synchronization in Immediate mode before you run the [niDCPower_MeasureMultiple](#) function. To force a synchronization in Immediate mode, set the [NIDCPOWER_ATTR_RESET_AVERAGE_BEFORE_MEASUREMENT](#) attribute to `VI_TRUE`, and then run the [niDCPower_MeasureMultiple](#) function specifying all channels in the **channelName** parameter. You can set the [NIDCPOWER_ATTR_RESET_AVERAGE_BEFORE_MEASUREMENT](#) attribute to `VI_FALSE` after the [niDCPower_MeasureMultiple](#) function completes.

NIDCPOWER_ATTR_RESET_AVERAGE_BEFORE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Channel	None	None

Description

Specifies whether the measurement returned from any measurement call starts with a new measurement call or returns a measurement that has already begun or completed.

When you set the [NIDCPOWER_ATTR_SAMPLES_TO_AVERAGE](#) attribute in [Immediate mode](#), the output channel measurements might move out of synchronization. While NI-DCPower automatically synchronizes measurements upon the initialization of a session, you can force a synchronization in [Immediate mode](#) before you run the [niDCPower_MeasureMultiple](#) function. To force a synchronization in [Immediate mode](#), set the

`NIDCPOWER_ATTR_RESET_AVERAGE_BEFORE_MEASUREMENT` attribute to `VI_TRUE`, and then run the [niDCPower_MeasureMultiple](#) function specifying all channels in the **channelName** parameter. You can set the

`NIDCPOWER_ATTR_RESET_AVERAGE_BEFORE_MEASUREMENT` attribute to `VI_FALSE` after the [niDCPower_MeasureMultiple](#) function completes.

Defined Values:

VI_TRUE	Reset the average before measurement.
VI_FALSE	Do not reset the average before measurement.

Default Value: VI_TRUE

NIDCPOWER_ATTR_OVERRANGING_ENABLED

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Device	None	None

Description

Specifies whether NI-DCPower allows setting the [voltage level](#), [current level](#), [voltage limit](#) and [current limit](#) outside the device's specification limits. Refer to the [Ranges](#) topic of the *NI DC Power Supplies and SMUs Help* for more information about overranging.

Defined Values:

VI_TRUE	Overranging is enabled.
VI_FALSE	Overranging is disabled.

Default Value: VI_FALSE

NIDCPOWER_ATTR_OUTPUT_CAPACITANCE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	R/W	Channel	None	None

Description

Selects a low or high capacitance on the output for the specified channel(s). Refer to the [Output Capacitance Selection](#) topic of the *NI DC Power Supplies and SMUs Help* for more information about capacitance.

Defined Values:

NIDCPOWER_VAL_LOW	Output capacitance is low.
NIDCPOWER_VAL_HIGH	Output capacitance is high.

NIDCPOWER_ATTR_RANGE_CHECK

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Device	None	niDCPower_InitWithOptions

Description

Specifies whether to validate attribute values and function parameters. If this attribute is enabled, NI-DCPower validates the parameter values that you pass to NI-DCPower functions. Range checking parameters is useful for debugging. After you validate your program, you can set this attribute to `VI_FALSE` to disable range checking and maximize performance.

The default value is `VI_TRUE`. Use the `niDCPower_InitWithOptions` function to override this value.

NIDCPOWER_ATTR_QUERY_INSTRUMENT_STA1

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Device	None	None

Description

Specifies whether NI-DCPower queries the device status after each operation. Querying the device status is useful for debugging. After you validate your program, you can set this attribute to `VI_FALSE` to disable status checking and maximize performance.

NI-DCPower ignores status checking for particular attributes regardless of the setting of this attribute. The default value is `VI_TRUE`. Use the `niDCPower_InitWithOptions` function to override this value.

NIDCPOWER_ATTR_CACHE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Device	None	niDCPower_InitWithOptions

Description

Specifies whether to cache the value of attributes. When caching is enabled, NI-DCPower records the current power supply settings and avoids sending redundant commands to the device. Enabling caching can significantly increase execution speed.

NI-DCPower might always cache or never cache particular attributes regardless of the setting of this attribute. The default value is VI_TRUE. Use the `niDCPower_InitWithOptions` function to override this value.

NIDCPOWER_ATTR_SIMULATE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Device	None	niDCPower_InitWithOptions

Description

Specifies whether to simulate NI-DCPower I/O operations.

Defined Values:

VI_TRUE	Simulate NI-DCPower I/O operations.
VI_FALSE	Do not simulate NI-DCPower I/O operations.

Default Value: VI_FALSE

NIDCPOWER_ATTR_RECORD_COERCIONS

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Device	None	None

Description

Specifies whether the IVI engine records the value coercions it makes for ViInt32 and ViReal64 attributes. Call the [niDCPower_GetNextCoercionRecord](#) function to read and delete the earliest coercion record from the list.

The default value is VI_FALSE. Use the [niDCPower_InitWithOptions](#) function to override this value.

NIDCPOWER_ATTR_INTERCHANGE_CHECK

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViBoolean	R/W	Device	None	niDCPower_InitWithOptions

Description

Specifies whether to perform interchangeability checking and log interchangeability warnings when you call NI-DCPower functions. The default value is VI_FALSE.

Interchangeability warnings indicate that using your application with a different power supply might cause different behavior. Call the [niDCPower_GetNextInterchangeWarning](#) function to retrieve interchange warnings.

Call the niDCPower_GetNextInterchangeWarning function to clear the list of interchangeability warnings without reading them.

Interchangeability checking examines the attributes in a capability group only if you specify a value for at least one attribute within that group. Interchangeability warnings can occur when an attribute affects the behavior of the device and you have not set that attribute or when the attribute has been invalidated since you set it.

NIDCPOWER_ATTR_SPECIFIC_DRIVER_PREFIX

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Contains the prefix for NI-DCPower. The name of each user-callable function in NI-DCPower begins with this prefix.

NIDCPOWER_ATTR_SPECIFIC_DRIVER_REVISIC

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Contains additional version information about NI-DCPower.

NIDCPOWER_ATTR_DRIVER_SETUP

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	niDCPower_InitWithOptions

Description

Indicates the Driver Setup string that the user specified when initializing the driver. Some cases exist where the user must specify instrument driver options at initialization time. An example of this is specifying a particular device model from among a family of devices that the driver supports. This is useful when [simulating](#) a device.

The user can specify driver-specific options through the DriverSetup keyword in the **optionsString** parameter in the [niDCPower_InitWithOptions](#) function. If the user does not specify a Driver Setup string, this attribute returns an empty string.

NIDCPOWER_ATTR_SUPPORTED_INSTRUMENT

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Contains a comma-separated list of supported power supply models.

NIDCPOWER_ATTR_GROUP_CAPABILITIES

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Contains a comma-separated list of class-extension groups that NI-DCPower implements.

NIDCPOWER_ATTR_CHANNEL_COUNT

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViInt32	RO	Device	None	None

Description

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

NIDCPOWER_ATTR_INSTRUMENT_MANUFACTU

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Contains the name of the manufacturer for the device you are currently using.

NIDCPOWER_ATTR_INSTRUMENT_MODEL

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Contains the model number or name of the device that you are currently using.

NIDCPOWER_ATTR_INSTRUMENT_FIRMWARE_I

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	niDCPower_revision_query

Description

Contains the firmware revision information for the device you are currently using.

NIDCPOWER_ATTR_LOGICAL_NAME

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Contains the logical name you specified when opening the current IVI session.

You can pass a logical name to the [niDCPower_init](#) or [niDCPower_InitWithOptions](#) functions. The IVI Configuration utility must contain an entry for the logical name. The logical name entry refers to a function section in the IVI Configuration file. The function section specifies a physical device and initial user options.

NIDCPOWER_ATTR_IO_RESOURCE_DESCRIPTOR

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	RO	Device	None	None

Description

Indicates the resource descriptor NI-DCPower uses to identify the physical device.

If you initialize NI-DCPower with a logical name, this attribute contains the resource descriptor that corresponds to the entry in the IVI Configuration utility.

If you initialize NI-DCPower with the resource descriptor, this attribute contains that value.

Examples

NI-DCPower examples are instructional tools that demonstrate power supply functionality. For example locations, refer to the *NI-DCPower Readme*.

niDCPower_revision_query

```
ViStatus niDCPower_revision_query (ViSession vi, ViChar  
    instrumentDriverRevision[], ViChar firmwareRevision[]);
```

Purpose

Returns the revision information of NI-DCPower and the device firmware.

Parameters

Input

Name	Type	Description
vi	ViSession	Identifies a particular instrument session. vi is obtained from the niDCPower_init or niDCPower_InitWithOptions function.

Output

Name	Type	Description
instrumentDriverRevision	ViChar[]	Returns the driver revision information for NI-DCPower.
firmwareRevision	ViChar[]	Returns firmware revision information for the device you are using. The size of this array must be at least 256 bytes.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--

Purpose

Creates a new IVI instrument driver session to the device specified in **resourceName** and returns a session handle you use to identify the device in all subsequent NI-DCPower function calls. This function also sends initialization commands to set the device to the state necessary for the operation of NI-DCPower.

To place the device in a known start-up state when creating a new session, set **resetDevice** to VI_TRUE. This action is equivalent to using the [niDCPower_reset](#) function.

To open a session and leave the device in its existing configuration without passing through a transitional output state, set **resetDevice** to VI_FALSE, and immediately call the [niDCPower_Abort](#) function. To apply a new configuration without disrupting the output channels of the device, configure the device in [Delayed Configuration](#) mode as in the previous session changing only the desired settings, and then call [niDCPower_Initiate](#) function.

Parameters

Input

Name	Type	Description
resourceName	ViRsrc	Specifies the resourceName assigned by Measurement & Automation Explorer (MAX), for example "PXI1Slot3" where "PXI1Slot3" is an instrument's resourceName . resourceName can also be a logical IVI name.
IDQuery	ViBoolean	Specifies whether the device is queried to determine if the device is a valid instrument for NI-DCPower.

Defined Values:

VI_TRUE (1)	Perform ID query.
VI_FALSE (0)	Do not perform ID query.

Default Value: VI_TRUE

resetDevice	ViBoolean	Specifies whether to reset the device during the initialization procedure.
--------------------	-----------	--

Defined Values:

VI_TRUE (1)	Reset the device.
VI_FALSE (0)	Do not reset the device.

Default Value: VI_TRUE

Output

Name	Type	Description
vi	ViSession	Returns a session handle that you use to identify the session in all subsequent NI-DCPower function calls.

Return Value

Name	Type	Description
------	------	-------------

Status	ViStatus	Reports the status of this operation. To obtain a text description of the status code, call the niDCPower_error_message function. To obtain additional information concerning the error condition, call the niDCPower_GetError function.
---------------	----------	--