

# NI-SWITCH Function Reference Help

June 2007, 372294A-01

Use the NI-SWITCH functions to create an application using LabWindows™/CVI™, Microsoft Visual C++, or Microsoft Visual Basic.

To comment on National Instruments documentation, refer to the [National Instruments Web site](#).

© 2007 National Instruments Corporation. All rights reserved.

# C/C++/VB Function Reference

Expand this book to view the C and VB functions and attributes for NI-SWITCH.

<b>Class/Panel Name</b>	<b>Function Name</b>
Initialize	<a href="#"><u>niSwitch_init</u></a>
Initialize With Options	<a href="#"><u>niSwitch_InitWithOptions</u></a>
Initialize With Topology	<a href="#"><u>niSwitch_InitWithTopology</u></a>
<b>Configuration Functions</b>	
<b>Set/Get/Check Attribute</b>	
<b>Set Attribute</b>	
Set Attribute ViInt32	<a href="#"><u>niSwitch_SetAttributeViInt32</u></a>
Set Attribute ViReal64	<a href="#"><u>niSwitch_SetAttributeViReal64</u></a>
Set Attribute ViString	<a href="#"><u>niSwitch_SetAttributeViString</u></a>
Set Attribute ViBoolean	<a href="#"><u>niSwitch_SetAttributeViBoolean</u></a>
Set Attribute ViSession	<a href="#"><u>niSwitch_SetAttributeViSession</u></a>
<b>Get Attribute</b>	
Get Attribute ViInt32	<a href="#"><u>niSwitch_GetAttributeViInt32</u></a>
Get Attribute ViReal64	<a href="#"><u>niSwitch_GetAttributeViReal64</u></a>
Get Attribute ViString	<a href="#"><u>niSwitch_GetAttributeViString</u></a>
Get Attribute ViBoolean	<a href="#"><u>niSwitch_GetAttributeViBoolean</u></a>
Get Attribute ViSession	<a href="#"><u>niSwitch_GetAttributeViSession</u></a>
<b>Check Attribute</b>	
Check Attribute ViInt32	<a href="#"><u>niSwitch_CheckAttributeViInt32</u></a>
Check Attribute ViReal64	<a href="#"><u>niSwitch_CheckAttributeViReal64</u></a>
Check Attribute ViString	<a href="#"><u>niSwitch_CheckAttributeViString</u></a>
Check Attribute ViBoolean	<a href="#"><u>niSwitch_CheckAttributeViBoolean</u></a>
Check Attribute ViSession	<a href="#"><u>niSwitch_CheckAttributeViSession</u></a>
<b>Route Functions</b>	

Connect Channels	<a href="#"><u>niSwitch_Connect</u></a>
Disconnect Channels	<a href="#"><u>niSwitch_Disconnect</u></a>
Disconnect All Channels	<a href="#"><u>niSwitch_DisconnectAll</u></a>
Switch Is Debounced?	<a href="#"><u>niSwitch_IsDebounced</u></a>
Wait For Debounce	<a href="#"><u>niSwitch_WaitForDebounce</u></a>
Can Connect Channels?	<a href="#"><u>niSwitch_CanConnect</u></a>
<b>Paths</b>	
Set Path	<a href="#"><u>niSwitch_SetPath</u></a>
Get Path	<a href="#"><u>niSwitch_GetPath</u></a>
<b>Scan Functions</b>	
Scan	<a href="#"><u>niSwitch_Scan</u></a>
Initiate Scan	<a href="#"><u>niSwitch_InitiateScan</u></a>
Abort Scan	<a href="#"><u>niSwitch_AbortScan</u></a>
Send Software Trigger	<a href="#"><u>niSwitch_SendSoftwareTrigger</u></a>
Switch Is Scanning?	<a href="#"><u>niSwitch_IsScanning</u></a>
Wait For Scan To Complete	<a href="#"><u>niSwitch_WaitForScanComplete</u></a>
Set Continuous Scan	<a href="#"><u>niSwitch_SetContinuousScan</u></a>
Configure Scanlist	<a href="#"><u>niSwitch_ConfigureScanList</u></a>
Configure Scan Trigger	<a href="#"><u>niSwitch_ConfigureScanTrigger</u></a>
Route Trigger Input	<a href="#"><u>niSwitch_RouteTriggerInput</u></a>
Route Scan Advanced Output	<a href="#"><u>niSwitch_RouteScanAdvancedOutput</u></a>
<b>Relay Operations</b>	
Get Relay Name	<a href="#"><u>niSwitch_GetRelayName</u></a>
Get Relay Count	<a href="#"><u>niSwitch_GetRelayCount</u></a>
Get Relay Position	<a href="#"><u>niSwitch_GetRelayPosition</u></a>
Relay Control	<a href="#"><u>niSwitch_RelayControl</u></a>
<b>Calibration Functions</b>	
Write Calibration Data	<a href="#"><u>niSwitch_CalibrationDataWrite</u></a>
Read Calibration Data	<a href="#"><u>niSwitch_CalibrationDataRead</u></a>

## Utility Functions

Commit	<a href="#"><u>niSwitch_Commit</u></a>
Get Channel Name	<a href="#"><u>niSwitch_GetChannelName</u></a>
Reset	<a href="#"><u>niSwitch_reset</u></a>
Reset With Defaults	<a href="#"><u>niSwitch_ResetWithDefaults</u></a>
Disable	<a href="#"><u>niSwitch_Disable</u></a>
Self-Test	<a href="#"><u>niSwitch_self_test</u></a>
Revision Query	<a href="#"><u>niSwitch_revision_query</u></a>
Error-Query	<a href="#"><u>niSwitch_error_query</u></a>
Error Message	<a href="#"><u>niSwitch_error_message</u></a>

## Coercion Info

Get Next Coercion Record	<a href="#"><u>niSwitch_GetNextCoercionRecord</u></a>
--------------------------	---

## Interchangeability Info

Get Next Interchange Warning	<a href="#"><u>niSwitch_GetNextInterchangeWarning</u></a>
Clear Interchange Warnings	<a href="#"><u>niSwitch_ClearInterchangeWarnings</u></a>
Reset Interchange Check	<a href="#"><u>niSwitch_ResetInterchangeCheck</u></a>

## Error Info

Get Error	<a href="#"><u>niSwitch_GetError</u></a>
Clear Error	<a href="#"><u>niSwitch_ClearError</u></a>

## Locking

Lock Session	<a href="#"><u>niSwitch_LockSession</u></a>
Unlock Session	<a href="#"><u>niSwitch_UnlockSession</u></a>
Close	<a href="#"><u>niSwitch_close</u></a>

# **niSwitch\_init**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_init (ViRsrc resourceName, ViBoolean idQuery,  
ViBoolean resetDevice, ViSession* vi);
```

## Purpose

Returns a session handle used to identify the switch module in all subsequent instrument driver calls.

[niSwitch\\_init](#) creates a new IVI instrument driver session for the switch module specified in the **resourceName** parameter. If multiple [topologies](#) are valid for that device, NI-SWITCH uses the default topology specified in MAX.

By default, the switch module is reset to a known state.

An error is returned if a session to the specified resource exists in another process. The same session is returned if [niSwitch\\_init](#) is called twice in the same process for the same resource with the same topology.

## Parameters

Name	Type	Description
resourceName	ViRsrc	Resource name of the switch module to initialize.

Syntax:

MAX	Configured under Valid Syntax
NI-DAQmx Devices	DAQmxDeviceName
Traditional NI-DAQ (Legacy) Devices	SCXI[chassis ID]::slot number
PXI System	PXI[bus number]::device number

Optional fields are shown in square brackets ([]). The default values for optional fields are as follows:

chassis ID = 1

bus number = 0



**Tip** IVI logical names are also valid for the resource name.

Example resource names:

Name	Description
SC1Mod3	NI-DAQmx module in chassis "SC1" Slot 3
MySwitch	NI-DAQmx module renamed to "MySwitch"
SCXI1::3	Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3
SCXI::3	Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3
PXI0::16	PXI bus 0, device number 16

PXI::16	PXI bus 0, device number 16
---------	-----------------------------

**idQuery** ViBoolean This parameter is ignored.  
 Because NI-SWITCH supports multiple switch modules, it always queries the switch to determine which device is installed. For this reason, this VI may return NISWITCH\_ERROR\_FAIL\_ID\_QUERY even if this parameter is set to [VI\\_FALSE](#).

Value	Description
<a href="#">VI_TRUE</a> (default)	Queries the switch to determine which device is installed.
<a href="#">VI_FALSE</a>	Currently unsupported.

**resetDevice** ViBoolean Specifies whether to reset the switch module during the initialization process.

Value	Description
<a href="#">VI_TRUE</a> (default)	Resets the device.
<a href="#">VI_FALSE</a>	Currently unsupported. The device will not reset.

**vi** ViSession A particular NI-SWITCH session established with [niSwitch\\_InitWithTopology](#), [niSwitch\\_InitWithOptions](#), or [niSwitch\\_init](#) and used for all subsequent NI-SWITCH calls.



# **niSwitch\_InitWithOptions**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_InitWithOptions (ViRsrc resourceName, ViBoolean idQuery,  
ViBoolean resetDevice, ViConstString optionString, ViSession* vi);
```

## Purpose

Returns a session handle used to identify the switch module in all subsequent instrument driver calls and optionally sets the initial state of the session.

[niSwitch\\_InitWithOptions](#) creates a new IVI instrument driver session for the switch module specified in the **resourceName** parameter. If multiple [topologies](#) are valid for that device, NI-SWITCH uses the default topology specified in MAX. The topology is also configurable in the **optionString** parameter.

By default, the switch module is reset to a known state.

Enable simulation in the **optionString** parameter.

An error is returned if a session to the specified resource exists in another process. The same session is returned if [niSwitch\\_InitWithOptions](#) is called twice in the same process for the same resource with the same topology.

## Parameters

**Name**                      **Type**  
**resourceName**   ViRsrc

### Description

Resource name of the switch module to identify the switch module.  
 Syntax:

MAX	Core
NI-DAQmx Devices	DAQ
Traditional NI-DAQ (Legacy) Devices	SCXI
PXI System	PXI

Optional fields are shown in square brackets. The following fields are as follows:

chassis ID = 1  
 bus number = 0



**Tip** IVI logical names are also valid.

Example resource names:

Name	Description
SC1Mod3	NI-DAQmx module in chassis 1
MySwitch	NI-DAQmx module renamed to MySwitch
SCXI1::3	Traditional NI-DAQ (Legacy) module 3 in chassis 1
SCXI::3	Traditional NI-DAQ (Legacy) module 3
PXI0::16	PXI bus 0, device number 16
PXI::16	PXI bus 0, device number 16

**idQuery**                      ViBoolean

This parameter is ignored.

Because NI-SWITCH supports multiple switch modules, you must specify a switch module to determine which device is installed. The error code NISWITCH\_ERROR\_FAIL\_ID\_QUERY is returned if the value is **VI\_FALSE**.

### Value

### Description

**VI\_TRUE** (default) Queries the switch to determine if the device is installed.

**VI\_FALSE** Currently unsupported.

<b>resetDevice</b>	ViBoolean	Specifies whether to reset the switch mod
		<b>Value</b>
		<b>Description</b>
		<a href="#"><u>VI_TRUE</u></a> (default) Reset device
		<a href="#"><u>VI_FALSE</u></a> Currently unsupported
<b>optionString</b>	ViConstString	Sets initial values of certain attributes for t table lists the attribute string names you c
		<b>Value</b>
		<a href="#"><u>NISWITCH_ATTR_RANGE_CHECK</u></a>
		<a href="#"><u>NISWITCH_ATTR_QUERY_INSTRUMEN</u></a>
		<a href="#"><u>NISWITCH_ATTR_CACHE</u></a>
		<a href="#"><u>NISWITCH_ATTR_SIMULATE</u></a>
		<a href="#"><u>NISWITCH_ATTR_RECORD_COERCION</u></a>
		<a href="#"><u>NISWITCH_ATTR_DRIVER_SETUP</u></a>

The format of the **optionString** is, "Attribute  
AttributeStringName is the name of the at  
value to which the attribute will be set. To  
assignments with a comma.

If you pass an empty string for this param  
default values for the attributes. You can c  
assigning a value. You do not have to spe  
do not specify an attribute, its default valu

Use the DriverSetup attribute to set the to  
or Traditional DAQ) of the switch module.  
token/value pairs within it.

DriverSetup=[config token]:[value];[config

Valid Config Tokens and Values:

<b>Value</b>	<b>Description</b>
--------------	--------------------

topology The topology of the device.  
for valid values.

resourcetype Use "daqmx" for devices co  
DAQmx Devices in MAX or  
configured under Traditional  
Devices in MAX.

For example, use the following string to se  
multiplexer configured in MAX under DAQ

"DriverSetup=topology:1127/2-Wire 32x1 Mu

The DriverSetup string is particularly impo  
the IviSwch class driver.

To enable simulation, set simulate equal to  
topology of the switch module to simulate.  
simulation for an NI SCXI-1127 configurec

"Simulate=1, DriverSetup=topology:1127/2-W

If simulate is set to 1 and the DriverSetup  
topology is used to determine which devic  
does not specify a topology, the device sp

vi

ViSession

A particular NI-SWITCH session establish  
[niSwitch\\_InitWithOptions](#), or [niSwitch\\_init](#) a  
SWITCH calls.

# **niSwitch\_InitWithTopology**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_InitWithTopology (ViRsrc resourceName,  
ViConstString topology, ViBoolean simulate, ViBoolean resetDevice,  
ViSession* vi);
```

## Purpose

Returns a session handle used to identify the switch module in all subsequent instrument driver calls and sets the [topology](#) of the switch module.

[niSwitch\\_InitWithTopology](#) creates a new IVI instrument driver session for the switch module specified in the `resourceName` parameter. The driver uses the topology specified in the **topology** parameter and overrides the topology specified in MAX.

By default, the switch module is reset to a known state.

Enable simulation by specifying the topology and setting the **simulate** parameter to [VI\\_TRUE](#).

## Parameters

Name	Type
resourceName	ViRsrc

### Description

Resource name of the switch module to initialize.

Syntax:

MAX	Configured under Valid Syntax
NI-DAQmx Devices	DAQmxDeviceName
Traditional NI-DAQ (Legacy) Devices	SCXI[chassis ID]::slot number
PXI System	PXI[bus number]::device number

Optional fields are shown in square brackets ([ ]). The default values for optional fields are as follows:

chassis ID = 1  
bus number = 0



**Tip** IVI logical names are also valid for the resource name.

Example resource names:

Name	Description
SC1Mod3	NI-DAQmx module in chassis "SC1" Slot 3
MySwitch	NI-DAQmx module renamed to "MySwitch"
SCXI1::3	Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3
SCXI::3	Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3
PXI0::16	PXI bus 0, device number 16



PXI::16 | PXI bus 0, device number 16

**topology** ViConstString Pass the topology name you want to use for the switch you specify with the **resourceName** parameter.



**Note** To determine the names of the supported topologies for your switch device, expand the **Devices** book, and select the switch module you are using from the **Contents** tab of this help file. In the device overview, the Operation Modes table(s) lists all supported topology and software names for the switch module.

**simulate** ViBoolean Enables simulation of the switch module specified in the **resourceName** parameter.

**Value**

**Description**

VI\_TRUE

Simulate

VI\_FALSE (default)

Do not simulate

**resetDevice** ViBoolean Specifies whether to reset the switch module during the initialization process.

**Value**

**Description**

VI\_TRUE  
(default)

Reset device

VI\_FALSE

The device will not reset.



**Note** The first call to niSwitch\_InitWithTopology, after you reboot your computer, will reset the hardware. This is the only case when the Reset flag is not honored.

**vi** ViSession A particular NI-SWITCH session

established with  
[niSwitch\\_InitWithTopology](#),  
[niSwitch\\_InitWithOptions](#), or [niSwitch\\_init](#)  
and used for all subsequent NI-SWITCH  
calls.

# **niSwitch\_close**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_close (ViSession vi);
```

## Purpose

Terminates the NI-SWITCH session and all of its attributes and deallocates any memory resources the driver uses.



**Note** You must unlock the session before calling [niSwitch\\_close](#). After calling [niSwitch\\_close](#), you cannot use the NI-SWITCH again until you call [niSwitch\\_init](#) or [niSwitch\\_InitWithOptions](#).

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_SetAttributeViInt32**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_SetAttributeViInt32 (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViInt32 attributeValue);
```

## **Purpose**

This function sets the value of a `ViInt32` attribute. This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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.

This instrument driver contains high-level functions that set most of the instrument 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, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute.  From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not



consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

### **attributeValue** ViInt32

Pass the value to which you want to set the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_SetAttributeViReal64**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_SetAttributeViReal64 (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViReal64 attributeValue);
```

## Purpose

This function sets the value of a ViReal64 attribute. This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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.

This instrument driver contains high-level functions that set most of the instrument 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, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute.  From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not

consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

### **attributeValue** ViReal64

Pass the value to which you want to set the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_SetAttributeViString**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_SetAttributeViString (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViConstString attributeValue);
```

## Purpose

Sets the value of a ViString attribute. You can use this low-level function to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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-SWITCH contains high-level functions that set most of the instrument 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, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute.  From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not



consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViConstString Pass the value to which you want to set the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_SetAttributeViBoolean**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_SetAttributeViBoolean (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViBoolean attributeValue);
```

## Purpose

Sets the value of a ViBoolean attribute. You can use this low-level function to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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-SWITCH contains high-level functions that set most of the instrument 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, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute.  From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not

consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViBoolean

Pass the value to which you want to set the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_SetAttributeViSession**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_SetAttributeViSession (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViSession attributeValue);
```

## Purpose

Sets the value of a ViSession attribute. You can use this as a low-level function to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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-SWITCH contains high-level functions that set most of the instrument 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, the functions check the instrument status after each call. Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute.  From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not



consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViSession

Pass the value to which you want to set the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_GetAttributeViInt32**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_GetAttributeViInt32 (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViInt32* attributeValue);
```

## **Purpose**

Queries the value of a ViInt32 attribute. You can use this function to get the values of instrument specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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>. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViInt32

Returns the current value of the attribute. Pass the address of a ViInt32 variable. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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>.

# **niSwitch\_GetAttributeViReal64**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_GetAttributeViReal64 (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViReal64* attributeValue);
```

## **Purpose**

Queries the value of a ViReal64 attribute. You can use this function to get the values of instrument specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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>. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViReal64

Returns the current value of the attribute. Pass the address of a ViReal64 variable. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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>.

# **niSwitch\_GetAttributeViString**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_GetAttributeViString (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViInt32 arraySize,  
ViChar[] attributeValue);
```

## Purpose

Queries the value of a ViString attribute. You can use this function to get the values of instrument specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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.

You must provide a ViChar array to serve as a buffer for the value. Pass the number of bytes in the buffer as **arraySize**. If the current value of the attribute, including the terminating NULL byte, is larger than the size you indicate in **arraySize**, the function copies **arraySize-1** bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the array size you must pass to get the entire value. For example, if the value is "123456" and **arraySize** is 4, the function places "123" into the buffer and returns 7. If you want to call this function just to get the required array size, you can pass 0 for **arraySize** and VI\_NULL for the **attributeValue** buffer. If you want the function to fill in the buffer regardless of the number of bytes in the value, pass a negative number for **arraySize**.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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>. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**arraySize** ViInt32

Pass the number of bytes in the ViChar array you specify for the Attribute Value parameter. If the current value of the attribute, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Array Size-1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the array size you must pass to get the entire value. For example, if the value is "123456" and the Array 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 the Attribute Value buffer parameter. The default value is 512.

**attributeValue** ViChar[]

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 the **arraySize** parameter. If the current value of the attribute, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies **arraySize-1** bytes into the buffer, places an ASCII NULL byte at the end of

the buffer, and returns the array size you must pass to get the entire value.

# **niSwitch\_GetAttributeViBoolean**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_GetAttributeViBoolean (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViBoolean* attributeValue);
```

## **Purpose**

Queries the value of a ViBoolean attribute. You can use this function to get the values of instrument specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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>. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViBoolean

Returns the current value of the attribute. Pass the address of a ViBoolean variable. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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>.

# **niSwitch\_GetAttributeViSession**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_GetAttributeViSession (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViSession* attributeValue);
```

## **Purpose**

Queries the value of a ViSession attribute. You can use this function to get the values of instrument specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument 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

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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>. A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViSession

Returns the current value of the attribute. Pass the address of a ViSession variable. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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>.

# **niSwitch\_CheckAttributeViInt32**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_CheckAttributeViInt32 (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViInt32 attributeValue);
```

## **Purpose**

Checks the validity of a value you specify for a VInt32 attribute.



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute.  From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not

consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

### **attributeValue** ViInt32

Pass the value which you want to verify as a valid value for the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_CheckAttributeViReal64**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_CheckAttributeViReal64 (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViReal64 attributeValue);
```

## **Purpose**

Checks the validity of a value you specify for a ViReal64 attribute.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViReal64 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

### **attributeValue** ViReal64

Pass the value which you want to verify as a valid value for the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_CheckAttributeViString**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_CheckAttributeViString (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViConstString attributeValue);
```

## **Purpose**

Checks the validity of a value you specify for a ViString attribute.



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViString type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViConstString Pass the value which you want to verify as a valid value for the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_CheckAttributeViBoolean**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_CheckAttributeViBoolean (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViBoolean attributeValue);
```

## **Purpose**

Checks the validity of a value you specify for a ViBoolean attribute.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViBoolean type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViBoolean

Pass the value which you want to verify as a valid value for the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_CheckAttributeViSession**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_CheckAttributeViSession (ViSession vi,  
ViConstString channelName, ViAttr attributeID, ViSession attributeValue);
```

## **Purpose**

Checks the validity of a value you specify for a ViSession attribute.



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Some attributes are unique for each channel. For these, pass the name of the channel. Other attributes are unique for each switch. Pass VI_NULL or an empty string for this parameter. The default value is an empty string.
<b>attributeID</b>	ViAttr	Pass the ID of an attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a 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 the ViSession type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

**attributeValue** ViSession

Pass the value which you want to verify as a valid value for the attribute. From the function panel window in LabWindows/CVI, you can use this control as follows. If the attribute currently showing in the Attribute ID 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 on the current settings of the instrument session.

# **niSwitch\_Connect**

## **IviSwtchBase Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_Connect (ViSession vi, ViConstString channel1,  
ViConstString channel2);
```

## Purpose

Creates a path between **channel1** and **channel2**. NI-SWITCH calculates and uses the shortest path between the two channels. Refer to [Immediate Operations](#) for information about channel usage types.

If a path is not available, the function returns one of the following errors:

- NISWITCH\_ERROR\_EXPLICIT\_CONNECTION\_EXISTS, if the two channels are already explicitly connected by calling either the [niSwitch\\_Connect](#) or [niSwitch\\_SetPath](#) function.
- NISWITCH\_ERROR\_IS\_CONFIGURATION\_CHANNEL, if a channel is a configuration channel. Error elaboration contains information about which of the two channels is a configuration channel.
- NISWITCH\_ERROR\_ATTEMPT\_TO\_CONNECT\_SOURCES, if both channels are connected to a different source. Error elaboration contains information about sources **channel1** and **channel2** connect to.
- NISWITCH\_ERROR\_CANNOT\_CONNECT\_TO\_ITSELF, if **channel1** and **channel2** are one and the same channel.
- NISWITCH\_ERROR\_PATH\_NOT\_FOUND, if the driver cannot find a path between the two channels.



**Note** Paths are bidirectional. For example, if a path exists between channels CH1 and CH2, then the path also exists between channels CH2 and CH1.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channel1</b>	ViConstString	Input one of the channel names of the desired path. Pass the other channel name as <b>channel2</b> . Refer to <a href="#">Devices</a> for valid channel names for the switch module.  Examples of valid channel names:  ch0, com0, ab0, r1, c2, cjtemp
<b>channel2</b>	ViConstString	Input one of the channel names of the desired path. Pass the other channel name as <b>channel1</b> . Refer to <a href="#">Devices</a> for valid channel names for the switch module.  Examples of valid channel names:  ch0, com0, ab0, r1, c2, cjtemp

# **niSwitch\_Disconnect**

## **IviSwchBase Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_Disconnect (ViSession vi, ViConstString channel1,  
ViConstString channel2);
```

## **Purpose**

Destroys the path between two channels that you create with the [niSwitch\\_Connect](#) or [niSwitch\\_SetPath](#) function.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channel1</b>	ViConstString	Input one of the channel names of the path to break. Pass the other channel name as <b>channel2</b> . Refer to <a href="#">Devices</a> for valid channel names for the switch module.  Examples of valid channel names:  ch0, com0, ab0, r1, c2, cjtemp
<b>channel2</b>	ViConstString	Input one of the channel names of the path to break. Pass the other channel name as <b>channel1</b> . Refer to <a href="#">Devices</a> for valid channel names for the switch module.  Examples of valid channel names:  ch0, com0, ab0, r1, c2, cjtemp



# **niSwitch\_DisconnectAll**

**IviSwchBase Capability Group**

## **C Function Prototype**

```
ViStatus niSwitch_DisconnectAll (ViSession vi);
```

## **Purpose**

Breaks all existing paths.

If the switch module cannot break all paths, the NISWITCH\_WARN\_PATH\_REMAINS warning is returned.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_IsDebounced**

## **IviSwchBase Capability Group**

### **C Function Prototype**

ViStatus niSwitch\_IsDebounced (ViSession vi, ViBoolean\* isDebounced);

## **Purpose**

Indicates if all created paths have settled by returning the value of the NISWITCH\_ATTR\_IS\_DEBOUNCED attribute.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>isDebounced</b>	ViBoolean	<a href="#">VI_TRUE</a> indicates that all created paths have settled. <a href="#">VI_FALSE</a> indicates that all created paths have not settled.

# **niSwitch\_WaitForDebounce**

## **IviSwchBase Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_WaitForDebounce (ViSession vi,  
ViInt32 maximumTime_ms);
```

## **Purpose**

Pauses until all created paths have settled.

If the time you specify with the **maximumTime\_ms** parameter elapses before the switch paths settle, this function returns the `NISWITCH_ERROR_MAX_TIME_EXCEEDED` error.



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>maximumTime_ms</b>	ViInt32	Specifies the maximum length of time to wait for all relays in the switch module to activate or deactivate. If the specified time elapses before all relays activate or deactivate, a timeout error is returned. The default value is 5000 ms.

# **niSwitch\_CanConnect**

## **IviSwtchBase Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_CanConnect (ViSession vi, ViConstString channel1,  
ViConstString channel2, ViInt32* pathCapability);
```

## **Purpose**

Verifies that a path between **channel1** and **channel2** can be created.

If a path is possible in the switch module, the availability of that path is returned given the existing connections. If the path is possible but in use, a NISWITCH\_WARN\_IMPLICIT\_CONNECTION\_EXISTS warning is returned.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established by <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_init</a> and used for all subsequent I/O.
<b>channel1</b>	ViConstString	Input one of the channel names of the device. Do not use the other channel name as the <b>channel2</b> . Refer to <a href="#">niSwitch_init</a> for valid channel names for the switch module. Examples of valid channel names: ch0, com0, ab0, r1, c2, cjtemp The default value is an empty string.
<b>channel2</b>	ViConstString	Input one of the channel names of the device. Do not use the other channel name as <b>channel1</b> . Refer to <a href="#">niSwitch_init</a> for valid channel names for the switch module. Examples of valid channel names: ch0, com0, ab0, r1, c2, cjtemp The default value is an empty string.
<b>pathCapability</b>	ViInt32	Indicates whether a path is valid. Possible values include:  <b>Value</b>  NISWITCH_VAL_PATH_AVAILABLE    NISWITCH_VAL_PATH_EXISTS

NISWITCH\_VAL\_PATH\_UNSUPPORTED

NISWITCH\_VAL\_RSRC\_IN\_USE

NISWITCH\_VAL\_SOURCE\_CONFLICT

NISWITCH\_VAL\_CHANNEL\_NOT\_AVAIL

## **niSwitch\_SetPath**

**IviSwchBase Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_SetPath (ViSession vi, ViConstString pathList);
```

## Purpose

Connects two channels by specifying an explicit path in **pathList**. [niSwitch\\_SetPath](#) is particularly useful where path repeatability is important, such as in calibrated signal paths. If this is not necessary, use [niSwitch\\_Connect](#).



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>pathList</b>	ViConstString	A string composed of comma-separated paths between channel 1 and channel 2. The first and last names in the path are the endpoints of the path. Every other channel in the path are configuration channels.

Example of a valid path list string:

ch0->com0, com0->ab0.

In this example, com0 is a configuration channel.

Obtain the path list for a previously created path with [niSwitch\\_GetPath](#).

# **niSwitch\_GetPath**

## **IviSwchBase Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_GetPath (ViSession vi, ViConstString channel1,  
ViConstString channel2, ViInt32 bufferSize, ViChar[] path);
```

## Purpose

Returns a string that identifies the explicit path created with [niSwitch\\_Connect](#). Pass this string to [niSwitch\\_SetPath](#) to establish the exact same path in future connections.

In some cases, multiple paths are available between two channels. When you call [niSwitch\\_Connect](#), NI-SWITCH selects an available path; however, the driver may not always select the same path through the switch module.

[niSwitch\\_GetPath](#) only returns those paths explicitly created by [niSwitch\\_Connect](#) or [niSwitch\\_SetPath](#). For example, if you connect channels CH1 and CH3, and then channels CH2 and CH3, an explicit path between channels CH1 and CH2 does not exist and an error is returned.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>channel1</b>	ViConstString	Input one of the channel names of the desired path. Pass the other channel name as <b>channel2</b> . Refer to <a href="#">Devices</a> for valid channel names for the switch module.  Examples of valid channel names: ch0, com0, ab0, r1, c2, cjtemp  The default value is an empty string.
<b>channel2</b>	ViConstString	Input one of the channel names of the desired path. Pass the other channel name as <b>channel1</b> . Refer to <a href="#">Devices</a> for valid channel names for the switch module.  Examples of valid channel names: ch0, com0, ab0, r1, c2, cjtemp  The default value is an empty string.
<b>bufferSize</b>	ViInt32	Pass the number of bytes in the ViChar array you specify for the <b>Path</b> parameter. If the current value of the attribute, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies <b>bufferSize</b> -1 bytes into the buffer, places an ASCII NULL 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 "R1->C1" and <b>bufferSize</b> is 4, the function places "R1-" into the buffer and returns 7. If you pass 0, you

can pass **VI\_NULL** for **path**. This enables you to find out the path size and to allocate the buffer of the appropriate size before calling this function again.

**path**

ViChar[]

A string composed of comma-separated paths between **channel1** and **channel2**. The first and last names in the path are the endpoints of the path. All other channels in the path are configuration channels.

Examples of returned paths:

ch0->com0, com0->ab0

# **niSwitch\_Scan**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_Scan (ViSession vi, ViConstString scanlist,  
ViInt16 initiation);
```

## Purpose

Takes the scan list provided, programs the switching hardware and initiates the scan. Once initiation is complete, the operation will return. The scan list itself is comprised of a list of channel connections separated by semicolons. For example, the following scan list would scan the first three channels of a multiplexer. Example: com0->ch0; com0->ch1; com0->ch2;. Refer to [scan lists](#) for additional information. To see the status of the scan, you can call either [niSwitch\\_IsScanning](#) or [niSwitch\\_WaitForScanComplete](#). Use the [niSwitch\\_ConfigureScanTrigger](#) function to configure the scan trigger. Use the [niSwitch\\_AbortScan](#) function to stop the scan if you are in continuous scan mode (Refer to [niSwitch\\_SetContinuousScan](#)); otherwise the scan halts automatically when the end of the scan list is reached. For reference, this operation is equivalent to calling [niSwitch\\_ConfigureScanList](#) and [niSwitch\\_InitiateScan](#).

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptic</a> used for all subsequent NI-SWITCH calls.

<b>scanlist</b>	ViConstString	Pass the <a href="#">scan list</a> you want the instrument to use.
<b>initiation</b>	ViInt16	Use the <b>initiation</b> parameter to specify whether measurement device initiates the scan trigger hardware parameter determines whether to wait for the scan point before completing. If the measurement device initiates the scan, set NISWITCH_VAL_MEASUREMENT_DEVICE_INITIATED then waits until the switch is waiting for a trigger device before completing. If the switch initiates the scan, set this parameter to NISWITCH_VAL_SWITCH_INITIATED. This function completes immediately after initiating the scan.

You should have already set up your DMM to wait for a scan point before calling this function with **Initiation** set to NISWITCH\_VAL\_SWITCH\_INITIATED.

### Value

NISWITCH\_VAL\_SWITCH\_INITIATED

NISWITCH\_VAL\_MEASUREMENT\_DEVICE\_INITIATED  
(default)



## **niSwitch\_InitiateScan**

**IviSwthScanner Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_InitiateScan (ViSession vi);
```

## Purpose

Commits the configured scan list and trigger settings to hardware and initiates the scan. If [niSwitch\\_Commit](#) was called earlier, [niSwitch\\_InitiateScan](#) only initiates the scan and returns immediately.

Once the scanning operation begins, you cannot perform any other operation other than `GetAttribute`, [niSwitch\\_AbortScan](#), or [niSwitch\\_SendSoftwareTrigger](#). All other functions return the `NISWITCH_ERROR_SCAN_IN_PROGRESS` error. To stop the scanning operation,

To stop the scanning operation, call [niSwitch\\_AbortScan](#).

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

## **niSwitch\_AbortScan**

**IviSwthScanner Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_AbortScan (ViSession vi);
```

## **Purpose**

Aborts the scan in progress.

Initiate a scan with [niSwitch\\_InitiateScan](#).

If the switch module is not scanning, the NISWITCH\_ERROR\_NO\_SCAN\_IN\_PROGRESS error is returned.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_SendSoftwareTrigger**

## **IviSwchSoftwareTrigger Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_SendSoftwareTrigger (ViSession vi);
```

## Purpose

Sends a software trigger to the switch specified in the NI-SWITCH session. When the trigger input is set to NISWITCH\_VAL\_SOFTWARE\_TRIG through either the [niSwitch\\_ConfigureScanTrigger](#) function or the [NISWITCH\\_ATTR\\_TRIGGER\\_INPUT](#) attribute, the scan does not proceed from a semicolon (wait for trigger) until [niSwitch\\_SendSoftwareTrigger](#) is called.



## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_IsScanning**

## **IviSwthScanner Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_IsScanning (ViSession vi, ViBoolean* isScanning);
```

## **Purpose**

Indicates the status of the scan.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>isScanning</b>	ViBoolean	NI-SWITCH returns the value of <a href="#">NISWITCH_ATTR_IS_SCANNING</a> attribute. <a href="#">VI_TRUE</a> indicates that the switch is scanning. <a href="#">VI_FALSE</a> indicates that the switch is idle.

# **niSwitch\_WaitForScanComplete**

## **IviSwthScanner Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_WaitForScanComplete (ViSession vi,  
ViInt32 maximumTime_ms);
```

## **Purpose**

Pauses until the switch stops scanning or until the maximum time has elapsed, when NI-SWITCH returns a timeout error.

If the time you specify with the **maximumTime\_ms** parameter elapsed before the scanning operation has finished, this function returns the `NISWITCH_ERROR_MAX_TIME_EXCEEDED` error.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> used for all subsequent NI-SWITCH calls
<b>maximumTime_ms</b>	ViInt32	Specifies the maximum length of time to for the switch module to stop scanning. If specified time elapses before the scan ends the <code>NISWITCH_ERROR_MAX_TIME_EXCEEDED</code> error is returned. The default value is 500

# **niSwitch\_SetContinuousScan**

## **IviSwthScanner Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_SetContinuousScan (ViSession vi,  
ViBoolean continuousScan);
```



## **Purpose**

Sets the to loop continuously through the scan list or to stop scanning after one pass through the scan list.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>continuousScan</b>	ViBoolean	If <a href="#">VI_TRUE</a> , loops continuously through the scan list during scanning. If <a href="#">VI_FALSE</a> , the scan stops after one pass through the scan list. The default value is <a href="#">VI_FALSE</a> .

# **niSwitch\_ConfigureScanList**

## **IviSwthScanner Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_ConfigureScanList (ViSession vi, ViConstString scanlist,  
ViInt32 scanMode);
```

## Purpose

Configures the scan list and scan mode used for scanning.

Refer to [Devices](#) to determine if the switch module supports scanning.

The scan list is comprised of a list of channel connections separated by semicolons. For example, the following scan list will scan the first three channels of a multiplexer:

```
com0->ch0; com0->ch1; com0->ch2;
```

Refer to [Scan Lists](#) for more information on scan list syntax.

To see the status of the scan, call either [niSwitch\\_IsScanning](#) or [niSwitch\\_WaitForScanComplete](#). Use the [niSwitch\\_ConfigureScanTrigger](#) function to configure the scan trigger. Use the [niSwitch\\_InitiateScan](#) function to start the scan.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>scanlist</b>	ViConstString	The scan list to use. NI-SWITCH uses this value to set the NISWITCH_ATTR_SCAN_LIST attribute.
<b>scanMode</b>	ViInt32	Specifies how the switch module breaks existing connections when scanning. The driver uses this value to set the NISWITCH_ATTR_SCAN_MODE attribute. Refer to <a href="#">scan modes</a> for more information. The default value is NISWITCH_VAL_BREAK_BEFORE_MAKE.

# **niSwitch\_ConfigureScanTrigger**

## **IviSwthScanner Capability Group**

### **C Function Prototype**

ViStatus niSwitch\_ConfigureScanTrigger (ViSession vi, ViReal64 scanDelay, ViInt32 triggerInput, ViInt32 scanAdvancedOutput);

## Purpose

Configures the scan triggers for the scan list established with [niSwitch\\_ConfigureScanList](#).

Refer to [Devices](#) to determine if the switch module supports scanning.

[niSwitch\\_ConfigureScanTrigger](#) sets the location that the switch expects to receive an input trigger to advance through the scan list. This function also sets the location where it outputs a scan advanced signal after it completes an entry in the scan list.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_i</a> used for all subsequent NI-SWITCH c
<b>scanDelay</b>	ViReal64	<p>The minimum length of time you want to wait after it creates a path until it as trigger on the scan advanced output li driver uses this value to set the <a href="#">NISWITCH_ATTR_SCAN_DELAY</a> attr scan delay is in addition to the settling driver uses this value to set the <a href="#">NISWITCH_ATTR_SCAN_DELAY</a> attri</p> <p>Express this value in seconds. The de is 0.0 s.</p>
<b>triggerInput</b>	ViInt32	Trigger source you want the switch m use during scanning. The driver uses to set the <a href="#">NISWITCH_ATTR_TRIGGEI</a> attribute. The switch waits for the trigg specify when it encounters a semicolc scan list. When the trigger occurs, the advances to the next entry in the scar to <a href="#">NISWITCH_ATTR_TRIGGER_INPU</a> of valid values.
<b>scanAdvancedOutput</b>	ViInt32	Output destination of the scan advanc signal. NI-SWITCH uses this value to <a href="#">NISWITCH_ATTR_SCAN_ADVANCEI</a> attribute. After the switch processes e in the scan list, it waits the length of ti specify in the <b>scanDelay</b> parameter a asserts a trigger on the line you speci parameter. Refer to



NISWITCH\_ATTR\_SCAN\_ADVANCEI  
for a list of valid values.

# **niSwitch\_RouteTriggerInput**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_RouteTriggerInput (ViSession vi,  
ViInt32 triggerInputConnector, ViInt32 triggerInputBusLine, ViBoolean invert);
```

## **Purpose**

Routes the input trigger from the front or rear connector to a trigger bus line (TTLx). To disconnect the route, call this function again and specify None for trigger bus line parameter.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
triggerInputConnector	ViInt32	The location of the input trigger source on the switch module. Valid locations are the <code>NISWITCH_VAL_FRONTCONNECTC</code> and <code>NISWITCH_VAL_REARCONNECTOF</code> . The default value is <code>NISWITCH_VAL_FRONTCONNECTC</code> .
triggerInputBusLine	ViInt32	The trigger line to route the input trigger. Select <code>NISWITCH_VAL_NON</code> to break an existing route.  Valid Values:  <code>NISWITCH_VAL_NONE</code> (default) <code>NISWITCH_VAL_TTL0</code> <code>NISWITCH_VAL_TTL1</code> <code>NISWITCH_VAL_TTL2</code> <code>NISWITCH_VAL_TTL3</code> <code>NISWITCH_VAL_TTL4</code> <code>NISWITCH_VAL_TTL5</code> <code>NISWITCH_VAL_TTL6</code> <code>NISWITCH_VAL_TTL7</code>
invert	ViBoolean	If <a href="#">VI_TRUE</a> , inverts the input trigger signal from falling to rising or vice versa. The default value is <a href="#">VI_FALSE</a> .

# **niSwitch\_RouteScanAdvancedOutput**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_RouteScanAdvancedOutput (ViSession vi,  
ViInt32 scanAdvancedOutputConnector, ViInt32 scanAdvancedOutputBusLine,  
ViBoolean invert);
```

## **Purpose**

Routes the scan advanced output trigger from a trigger bus line (TTLx) to the front or rear connector.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for subsequent NI-SWITCH calls.
<b>scanAdvancedOutputConnector</b>	ViInt32	The scan advanced output destination.  Valid Values:  NISWITCH_VAL_FRONTCONNECTOR (default) NISWITCH_VAL_REARCONNECTOR
<b>scanAdvancedOutputBusLine</b>	ViInt32	The trigger line to route the advanced output trigger front or rear connector. Select NISWITCH_VAL_NONE to use an existing route.  Valid Values:  NISWITCH_VAL_NONE (current) NISWITCH_VAL_TTL0 NISWITCH_VAL_TTL1 NISWITCH_VAL_TTL2 NISWITCH_VAL_TTL3 NISWITCH_VAL_TTL4 NISWITCH_VAL_TTL5 NISWITCH_VAL_TTL6 NISWITCH_VAL_TTL7

**invert**

ViBoolean If VI\_TRUE, inverts the input signal from falling to rising or rising to falling, versa. The default value is



# **niSwitch\_GetRelayName**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_GetRelayName (ViSession vi, ViInt32 index,  
ViInt32 relayNameBufferSize, ViChar[] relayNameBuffer);
```

## Purpose

Returns the relay name string that is in the relay list at the specified index.

Use [niSwitch\\_GetRelayName](#) in a For Loop to get a complete list of valid relay names for the switch. Use the NISWITCH\_ATTR\_NUMBER\_OF\_RELAYS attribute to determine the number of relays.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>index</b>	ViInt32	A 1-based index into the channel table. The default value is 1. The maximum value is the value of the NISWITCH_ATTR_CHANNEL_COUNT attribute.
<b>relayNameBufferSize</b>	ViInt32	Pass the number of bytes in the ViChan array you specify for the <b>relayNameBuffer</b> parameter. If the relay name string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NULL 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 <b>relayBufferSize</b> 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 the Coercion Record buffer parameter.

**relayNameBuffer**

ViChar[]

Returns the relay name for the index you specify.

# **niSwitch\_GetRelayCount**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_GetRelayCount (ViSession vi, ViConstString relayName,  
ViInt32* relayCount);
```

## Purpose

Returns the number of times the relay has changed from closed to open. Relay count is useful for tracking relay lifetime and usage. Call [niSwitch\\_WaitForDebounce](#) before `niSwitch_GetRelayCount` to ensure an accurate count.

Refer to [Devices](#) to determine if the switch module supports [individual relay control](#).

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>relayName</b>	ViConstString	Name of the relay. Refer to <a href="#">Devices</a> for a list of valid relay names for the switch module.  Examples of valid relay names:  ch0, ab0, 1wire, hlselect
<b>relayCount</b>	ViInt32	The number of relay cycles.

# **niSwitch\_GetRelayPosition**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_GetRelayPosition (ViSession vi, ViConstString relayName,  
ViInt32* relayPosition);
```



## **Purpose**

Returns the relay position for the relay specified in the **relayName** parameter.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>relayName</b>	ViConstString	Name of the relay. Refer to <a href="#">Devices</a> for a list of valid relay names for the switch module.  Examples of valid relay names:  ch0, ab0, 1wire, hlselect
<b>relayPosition</b>	ViInt32	Indicates whether the relay is open or closed.  Valid Values:  NISWITCH_VAL_OPEN (10) NISWITCH_VAL_CLOSED (11)

# **niSwitch\_RelayControl**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_RelayControl (ViSession vi, ViConstString relayName,  
ViInt32 relayAction);
```

## **Purpose**

Controls individual relays of the switch. When controlling individual relays, the protection offered by setting the usage of source channels and configurations channels is void.

Refer to [Devices](#) to determine if the switch module supports [individual relay control](#).

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>relayName</b>	ViConstString	Name of the relay. Refer to <a href="#">Devices</a> for a list of valid relay names for the switch module.

Examples of valid relay names:

ch0, ab0, 1wire, hlselect

<b>relayAction</b>	ViInt32	Specifies whether to open or close a given relay.
--------------------	---------	---

Defined values:

NISWITCH\_VAL\_OPEN\_RELAY

NISWITCH\_VAL\_CLOSE\_RELAY (default).

# **niSwitch\_Commit**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_Commit (ViSession vi);
```

## Purpose

Downloads the configured scan list and trigger settings to hardware.

Calling [niSwitch\\_Commit](#) is optional as it is implicitly called during [niSwitch\\_InitiateScan](#). Use [niSwitch\\_Commit](#) to arm triggers in a given order or to control when expensive hardware operations are performed.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.



# **niSwitch\_GetChannelName**

## **IviSwtchBase Capability Group**

### **C Function Prototype**

```
ViStatus niSwitch_GetChannelName (ViSession vi, ViInt32 index,  
ViInt32 bufferSize, ViChar[] channelNameBuffer);
```

## **Purpose**

Returns the channel string that is in the channel table at the specified index.

Use [niSwitch\\_GetChannelName](#) in a For Loop to get a complete list of valid channel names for the switch. Use the NISWITCH\_ATTR\_CHANNEL\_COUNT attribute to determine the number of channels.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>index</b>	ViInt32	A 1-based index into the channel table. The default value is 1. The maximum value is Value of Channel Count attribute.
<b>bufferSize</b>	ViInt32	Pass the number of bytes in the ViChar array you specify for the Channel Name Buffer parameter. If the channel name string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NULL 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 <b>bufferSize</b> 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 the Coercion Record buffer parameter.
<b>channelNameBuffer</b>	ViChar[]	Returns the channel name that is in

the channel table at the index you specify.

# **niSwitch\_reset**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_reset (ViSession vi);
```

## **Purpose**

Disconnects all created paths and returns the switch module to the state at initialization. Configuration channel and source channel settings remain unchanged.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_ResetWithDefaults**

## **Specific Function**

## **C Function Prototype**

ViStatus niSwitch\_ResetWithDefaults (ViSession vi);



## **Purpose**

Resets the switch module and applies initial user specified settings from the logical name used to initialize the session. If the session was created without a logical name, this function is equivalent to [niSwitch\\_reset](#).

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_Disable**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_Disable (ViSession vi);
```

## **Purpose**

Places the switch module in a quiescent state, where it has minimal or no impact on the system to which it is connected. All channels are disconnected and any scan in progress is aborted.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_self\_test**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_self_test (ViSession vi, ViInt16* selfTestResult,  
ViChar[] selfTestMessage);
```

## **Purpose**

Verifies that NI-SWITCH can communicate with the switch.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>selfTestResult</b>	ViInt16	Value returned from the switch self-test.  <b>0</b> Passed <b>1</b> Failed
<b>selfTestMessage</b>	ViChar[]	Self-test response string from the switch. You must pass a ViChar array with at least 256 bytes.



# **niSwitch\_revision\_query**

## **Specific Function**

## **C Function Prototype**

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

## **Purpose**

Returns the revision of the NI-SWITCH driver.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>instrumentDriverRevision</b>	ViChar[]	NI-SWITCH software revision numbers in the form of a string. You must pass a ViChar array with at least 256 bytes.
<b>firmwareRevision</b>	ViChar[]	Currently unsupported.

# **niSwitch\_error\_query**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_error_query (ViSession vi, ViInt32* errorCode,  
ViChar[] errorMessage);
```

## **Purpose**

This function reads an error code and a message from the instrument error queue.

NI-SWITCH does not have an error queue, so this function never returns any errors.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>errorCode</b>	ViInt32	Returns the error code read from the instrument error queue. NI-SWITCH does not have an error queue, so this function never returns any errors.
<b>errorMessage</b>	ViChar[]	Returns the error message string read from the instrument's error message queue. You must pass a ViChar array with at least 256 bytes. NI-SWITCH does not have an error queue, so this function only returns <b>No error</b> .

# **niSwitch\_error\_message**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_error_message (ViSession vi, ViStatus errorCode,  
ViChar[] errorMessage);
```

## **Purpose**

Converts an error code returned by NI-SWITCH into a user-readable string. Generally this information is supplied in error out of any NI-SWITCH VI. Use [niSwitch\\_error\\_message](#) for a static lookup of an error code description.



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>errorCode</b>	ViStatus	Status code returned by any NI-SWITCH function. The default value is 0 (VI_SUCCESS).
<b>errorMessage</b>	ViChar[]	The error information formatted into a string. You must pass a ViChar array with at least 256 bytes.

# **niSwitch\_GetNextCoercionRecord**

## **Specific Function**

### **C Function Prototype**

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

## Purpose

This function returns the coercion information associated with the IVI session. This function retrieves and clears the oldest instance in which NI-SWITCH coerced a value you specified to another value.

If you set the [NISWITCH\\_ATTR\\_RECORD\\_COERCIONS](#) attribute to [VI\\_TRUE](#), NI-SWITCH keeps a list of all coercions it makes on ViInt32 or ViReal64 values you pass to NI-SWITCH functions. You use this function to retrieve information from that list. If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies **bufferSize**-1 bytes into the buffer, places an ASCII NULL 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 **bufferSize** 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 the Coercion Record buffer parameter. The function returns an empty string in the Coercion Record parameter if no coercion records remain for the session.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>bufferSize</b>	ViInt32	Pass the number of bytes in the ViChar array you specify for the Coercion Record parameter. If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies <b>bufferSize</b> –1 bytes into the buffer, places an ASCII NULL 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 <b>bufferSize</b> 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 the Coercion Record buffer parameter.
<b>coercionRecord</b>	ViChar[]	Returns the next coercion record for the IVI session. If there are no coercion records, the function returns an empty string. The buffer must contain at least as many elements as the value you specify with the <b>bufferSize</b> parameter. If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate with the <b>bufferSize</b>

parameter, the function copies **bufferSize**–1 bytes into the buffer, places an ASCII NULL 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 **bufferSize** is 4, the function places "123" into the buffer and returns 7. This parameter returns an empty string if no coercion records remain for the session.

# **niSwitch\_GetNextInterchangeWarning**

## **Specific Function**

## **C Function Prototype**

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

## Purpose

This function returns the interchangeability warnings associated with the IVI session. It retrieves and clears the oldest instance in which the class driver recorded an interchangeability warning. Interchangeability warnings indicate that using your application with a different instrument might cause different behavior. You use this function to retrieve interchangeability warnings. The driver performs interchangeability checking when the NISWITCH\_ATTR\_INTERCHANGE\_CHECK attribute is set to VI\_TRUE. The function returns an empty string in the **interchangeWarning** parameter if no interchangeability warnings remain for the session. In general, the instrument driver generates interchangeability warnings when an attribute that affects the behavior of the instrument is in a state that you did not specify.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>bufferSize</b>	ViInt32	Pass the number of bytes in the ViChar array you specify for the <b>interchangeWarning</b> parameter. If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies <b>bufferSize</b> –1 bytes into the buffer, places an ASCII NULL 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 <b>bufferSize</b> 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 the Interchange Warning buffer parameter.
<b>interchangeWarning</b>	ViChar[]	Returns the next interchange warning for the IVI session. If there are no interchange warnings, the function returns an empty string. The buffer must contain at least as many elements as the value you specify with



the **bufferSize** parameter. If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate with the **bufferSize** parameter, the function copies **bufferSize**-1 bytes into the buffer, places an ASCII NULL 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 **bufferSize** is 4, the function places "123" into the buffer and returns 7. This parameter returns an empty string if no interchangeability warnings remain for the session.

# **niSwitch\_ClearInterchangeWarnings**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_ClearInterchangeWarnings (ViSession vi);
```

## **Purpose**

This function clears the list of current interchange warnings.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_ResetInterchangeCheck**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_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, ensure 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 instrument state depends on the configuration from a previously executed test module. Therefore, 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 behavior change 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 [niSwitch\\_GetNextInterchangeWarning](#) function only returns those interchangeability warnings that are generated after calling this function, you must clear the list of interchangeability warnings by repeatedly calling the [niSwitch\\_GetNextInterchangeWarning](#) function until no interchangeability warnings are returned. If you are not interested in the content of those warnings, you can call the [niSwitch\\_ClearInterchangeWarnings](#) function.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_GetError**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_GetError (ViSession vi, ViStatus* code, ViInt32 buffersize,  
ViChar[] description);
```



## Purpose

This function retrieves and then clears the IVI error information for the session or the current execution thread.

One exception exists: If the **bufferSize** parameter is 0, the function does not clear the error information. By passing 0 for the buffer size, the caller can ascertain the buffer size required to get the entire error description string and then call the function again with a sufficiently large buffer. If you specify a valid IVI session for the **vi** parameter, this function retrieves and then clears the error information for the session. If the user passes VI\_NULL for the **vi** parameter, this function retrieves and then clears the error information for the current execution thread. If the **vi** parameter 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 [niSwitch\\_GetError](#) or [niSwitch\\_ClearError](#).

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>code</b>	ViStatus	Returns the error code for the session or execution thread. If you pass 0 for <b>bufferSize</b> , you can pass VI_NULL for this parameter.
<b>bufferSize</b>	ViInt32	Pass the number of bytes in the ViChar array you specify for the Description parameter.  If the error description, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies <b>bufferSize</b> –1 bytes into the buffer, places an ASCII NULL 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 <b>bufferSize</b> 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 the <b>description</b> buffer parameter.
<b>description</b>	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 the <b>bufferSize</b> parameter. If the error

description, including the terminating NULL byte, contains more bytes than you indicate with the **bufferSize**, the function copies **bufferSize**–1 bytes into the buffer, places an ASCII NULL 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 **bufferSize** is 4, the function places "123" into the buffer and returns 7. If you pass 0 for the Buffer Size, you can pass VI\_NULL for this parameter.

# **niSwitch\_ClearError**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_ClearError (ViSession vi);
```

## Purpose

This function clears the error code and error description for the IVI session.

If you specify a valid IVI session for the **vi** parameter, this function clears the error information for the session.

If the user passes VI\_NULL for the **vi** parameter, this function clears the error information for the current execution thread.

If **vi** 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 deallocates 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 [niSwitch\\_GetError](#) function, which occurs when a call to [niSwitch\\_init](#) or [niSwitch\\_InitWithOptions](#) fails.

## Parameters

Name	Type	Description
vi	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.

# **niSwitch\_LockSession**

## **Specific Function**

## **C Function Prototype**

ViStatus niSwitch\_LockSession (ViSession vi, ViBoolean\* callerHasLock);

## Purpose

This function obtains a multithread lock on the instrument session. Before it does so, it waits until all other execution threads have released their locks on the instrument session. Other threads might have obtained a lock on this session in the following ways:

- Your application called [niSwitch\\_LockSession](#).
- A call to the instrument driver locked the session.
- A call to the IVI engine locked the session.

After your call to [niSwitch\\_LockSession](#) returns successfully, no other threads can access the instrument session until you call [niSwitch\\_UnlockSession](#). Use [niSwitch\\_LockSession](#) and [niSwitch\\_UnlockSession](#) around a sequence of calls to NI-SWITCH functions if you require that the instrument retain its settings through the end of the sequence. You can safely make nested calls to [niSwitch\\_LockSession](#) within the same thread. To completely unlock the session, balance each call to [niSwitch\\_LockSession](#) with a call to [niSwitch\\_UnlockSession](#). If, however, you use the **callerHasLock** parameter in all calls to [niSwitch\\_LockSession](#) and [niSwitch\\_UnlockSession](#) within a function, the IVI Library locks the session only once within the function regardless of the number of calls you make to [niSwitch\\_LockSession](#). This allows you to call [niSwitch\\_UnlockSession](#) just once at the end of the function.



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>callerHasLock</b>	ViBoolean	<p>This parameter serves as a convenience. If you do not want to use this parameter, pass <code>VI_NULL</code>.</p> <p>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 <a href="#">VI_FALSE</a>. Pass the address of the same local variable to any other calls you make to <a href="#">niSwitch_LockSession</a> or <a href="#">niSwitch_UnlockSession</a> in the same function. The parameter is an input/output parameter. <a href="#">niSwitch_LockSession</a> and <a href="#">niSwitch_UnlockSession</a> each inspect the current value and take the following actions:</p> <ul style="list-style-type: none"><li>• If the value is <a href="#">VI_TRUE</a>, <a href="#">niSwitch_LockSession</a> does not lock the session again. If the value is <a href="#">VI_FALSE</a>, <a href="#">niSwitch_LockSession</a> obtains the lock and sets the value of the parameter to <a href="#">VI_TRUE</a>.</li><li>• If the value is <a href="#">VI_FALSE</a>, <a href="#">niSwitch_UnlockSession</a> does not attempt to unlock the session. If the value is <a href="#">VI_TRUE</a>, <a href="#">niSwitch_UnlockSession</a> releases the lock and sets the value of the parameter to <a href="#">VI_FALSE</a>.</li></ul> <p>Thus, you can, call</p>

[niSwitch\\_UnlockSession](#) at the end of your function without worrying about whether you actually have the lock.

Example:

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
{
ViStatus error = VI_SUCCESS;
ViBoolean haveLock = VI_FALSE;
if (flags & BIT_1)
{
viCheckErr( niSwitch\_LockSession(vi,
&haveLock));
viCheckErr( TakeAction1(vi));
if (flags & BIT_2)
{
viCheckErr( niSwitch\_UnlockSession(vi,
&haveLock));
viCheckErr( TakeAction2(vi));
viCheckErr( niSwitch\_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. */
```

```
niSwitch\_UnlockSession(vi, &haveLock);
return error;
}
```

# **niSwitch\_UnlockSession**

## **Specific Function**

## **C Function Prototype**

```
ViStatus niSwitch_UnlockSession (ViSession vi, ViBoolean* callerHasLock);
```

## Purpose

This function releases a lock that you acquired on an instrument session using [niSwitch\\_LockSession](#).

Refer to [niSwitch\\_LockSession](#) for additional information on session locks.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established with <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> and used for all subsequent NI-SWITCH calls.
<b>callerHasLock</b>	ViBoolean	This parameter serves as a convenience. If you do not want to use this parameter, pass <code>VI_NULL</code> . 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 <a href="#">VI_FALSE</a> . Pass the address of the same local variable to any other calls you make to <a href="#">niSwitch_LockSession</a> or <a href="#">niSwitch_UnlockSession</a> in the same function. The parameter is an input/output parameter. <a href="#">niSwitch_LockSession</a> and <a href="#">niSwitch_UnlockSession</a> each inspect the current value and take the following actions: <ul style="list-style-type: none"><li>- If the value is <a href="#">VI_TRUE</a>, <a href="#">niSwitch_LockSession</a> does not lock the session again. If the value is <a href="#">VI_FALSE</a>, <a href="#">niSwitch_LockSession</a> obtains the lock and sets the value of the parameter to <a href="#">VI_TRUE</a>.</li><li>- If the value is <a href="#">VI_FALSE</a>, <a href="#">niSwitch_UnlockSession</a> does not attempt to unlock the session. If the value is <a href="#">VI_TRUE</a>, <a href="#">niSwitch_UnlockSession</a> releases the lock and sets the value of the parameter to <a href="#">VI_FALSE</a>.</li></ul>

Thus, you can, call [niSwitch\\_UnlockSession](#) at the end of your function without worrying about whether you actually have the lock.

Example:

```
ViStatus TestFunc (ViSession vi, ViInt32  
flags)
```

```
{  
ViStatus error = VI_SUCCESS;  
ViBoolean haveLock = VI_FALSE;  
if (flags & BIT_1)  
{  
viCheckErr( niSwitch\_LockSession(vi,  
&haveLock));  
viCheckErr( TakeAction1(vi)); if (flags &  
BIT_2)  
{  
viCheckErr( niSwitch\_UnlockSession(vi,  
&haveLock));  
viCheckErr( TakeAction2(vi));  
viCheckErr( niSwitch\_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. */
```

```
niSwitch\_UnlockSession(vi, &haveLock);  
return error;  
}
```

# **niSwitch\_CalibrationDataRead**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_CalibrationDataRead (ViSession vi,  
ViConstString channelName, ViInt32 calibrationField,  
ViReal64* calibrationData, ViInt32* calibrationDate_Year,  
ViInt32* calibrationDate_Month, ViInt32* calibrationDate_Day);
```

## **Purpose**

Retrieves the calibration data, typically in terms of the amplifier offset, stored in the EEPROM.

Some NI switches have an amplifier that may require periodic calibrations. You can perform the necessary calibration and store the data locally on the switch module EEPROM. The calibration date is also stored in the EEPROM.



## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session (created by <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_InitWithDefaultOptions</a> ) for all subsequent NI-SWITCH calls.
<b>channelName</b>	ViConstString	Name of the channel calibrated.  Examples of valid channel names:  ch0, com0, ab0, r1, c2, cjtemp  Refer to <a href="#">Devices</a> for a complete list of valid channel names.  While <a href="#">niSwitch_ReadCalibrationData</a> and <a href="#">niSwitch_WriteCalibrationData</a> take a channel name, some switches only support calibration for all input channels. NI PXI-2501 uses an optional set of input channels ch0 through ch47 to determine calibration time. In these cases, writing a different channel causes the previous calibration to be overwritten. Therefore, reading data from a different channel returns the same calibration data.
<b>calibrationField</b>	ViInt32	Tells NI-SWITCH which particular calibration parameter is associated with this channel. Valid values depend on the switch.  Examples of possible values: NISWITCH_VAL_CALIBRATION_CURRENT, NISWITCH_VAL_CALIBRATION_VOLTAGE, NISWITCH_VAL_CALIBRATION_TEMPERATURE, (1)

<b>calibrationData</b>	ViReal64	Calibration data from the EEPRC
<b>calibrationDate_Year</b>	ViInt32	Year the switch was last calibrate calibration data. For example, the 2003 would be returned as <b>2003</b> l
<b>calibrationDate_Month</b>	ViInt32	Month the switch was last calibra calibration data. For example, the 2003 would be returned as <b>8</b> by t
<b>calibrationDate_Day</b>	ViInt32	Day the switch was last calibrate calibration data. For example, the 2003 would be returned as <b>1</b> by t

# **niSwitch\_CalibrationDataWrite**

## **Specific Function**

### **C Function Prototype**

```
ViStatus niSwitch_CalibrationDataWrite (ViSession vi,  
ViConstString channelName, ViInt32 calibrationField,  
ViReal64 calibrationData);
```

## **Purpose**

Writes the calibration data, typically in terms of the amplifier offset, in the EEPROM.

Some NI switches have an amplifier that may require periodic calibrations. You can perform the necessary calibration and store the data locally on the switch EEPROM. The calibration date is also stored in the EEPROM.

## Parameters

Name	Type	Description
<b>vi</b>	ViSession	A particular NI-SWITCH session established by <a href="#">niSwitch_InitWithTopology</a> , <a href="#">niSwitch_InitWithOptions</a> , or <a href="#">niSwitch_init</a> for all subsequent NI-SWITCH calls.

<b>channelName</b>	ViConstString	Name of the channel calibrated.
--------------------	---------------	---------------------------------

Examples of valid channel names:

ch0, com0, ab0, r1, c2, cjtemp

Refer to [Devices](#) for a complete list of valid channel names.

While [niSwitch\\_ReadCalibrationData](#) and [niSwitch\\_WriteCalibrationData](#) take a channel name, some switches only support a single calibration for all input channels. For example, NI PXI-2501 uses an optional single amp channel ch0 through ch47 to decrease settling time. In these cases, writing a different value to a different channel causes the previous value to be overwritten. Therefore, reading different channels returns the same calibration data.

<b>calibrationField</b>	ViInt32	Tells NI-SWITCH which particular calibration parameter associated with this channel to use. Valid values depend on the switch hardware.
-------------------------	---------	---

Examples of possible values:

NISWITCH\_VAL\_CALIBRATION\_CJS\_A1  
NISWITCH\_VAL\_CALIBRATION\_CHANNEL1  
(1)

**calibrationData** ViReal64

Calibration data to store in the EEPROM

# **NISWITCH\_VAL\_BREAK\_AFTER\_MAKE**

## **Description**

When scanning, the switch breaks existing connections after making new connections.

## Defined Value

2



# **NISWITCH\_VAL\_BREAK\_BEFORE\_MAKE**

## **Description**

When scanning, the switch breaks existing connections before making new connections.

# Defined Value

1

# **NISWITCH\_VAL\_EXTERNAL**

## **Description**

External Trigger. The switch waits until it receives a trigger from an external source through the external trigger input before processing the next entry in the scan list.

## Defined Value

2

# **NISWITCH\_VAL\_FALLING\_EDGE**

## **Description**

The trigger occurs on the falling edge of the signal.

# Defined Value

1

# **NISWITCH\_VAL\_FRONTCONNECTOR**

## **Description**

The switch waits until it receives a trigger on the front connector.

**Defined Value**

1001



# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE1**

## **Description**

The switch waits until it receives a trigger on the front connector module 1.

## Defined Value

1041

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE1**

## **Description**

The switch waits until it receives a trigger on the front connector module 10.

## Defined Value

1050

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE1**

## **Description**

The switch waits until it receives a trigger on the front connector module 11.

## Defined Value

1051

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE1**

## **Description**

The switch waits until it receives a trigger on the front connector module 12.

## Defined Value

1052



# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE2**

## **Description**

The switch waits until it receives a trigger on the front connector module 2.

## Defined Value

1042

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE3**

## **Description**

The switch waits until it receives a trigger on the front connector module 3.

## Defined Value

1043

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE4**

## **Description**

The switch waits until it receives a trigger on the front connector module 4.

## Defined Value

1044

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE5**

## **Description**

The switch waits until it receives a trigger on the front connector module 5.

## Defined Value

1045



# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE6**

## **Description**

The switch waits until it receives a trigger on the front connector module 6.

## Defined Value

1046

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE7**

## **Description**

The switch waits until it receives a trigger on the front connector module 7.

## Defined Value

1047

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE8**

## **Description**

The switch waits until it receives a trigger on the front connector module 8.

## Defined Value

1048

# **NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE9**

## **Description**

The switch waits until it receives a trigger on the front connector module 9.

## Defined Value

1049



# **NISWITCH\_VAL\_IMMEDIATE**

## **Description**

Immediate Trigger. The switch does not wait for a trigger before processing the next entry in the scan list.

# Defined Value

1

# **NISWITCH\_VAL\_MASTER**

## **Description**

Multiple switches are sharing bused trigger lines for the scan and this device is the trigger master. You must set NISWITCH\_ATTR\_MASTER\_SLAVE\_TRIGGER\_BUS, NISWITCH\_ATTR\_MASTER\_SLAVE\_SCAN\_ADVANCED\_BUS, NISWITCH\_ATTR\_SCAN\_ADVANCED\_OUTPUT and NISWITCH\_ATTR\_TRIGGER\_INPUT for this device.

# Defined Value

1

# **NISWITCH\_VAL\_NONE**

## **Description**

No implicit action on connections when scanning.

**Defined Value**

0

## **NISWITCH\_VAL\_PXI\_STAR**

### **Description**

The switch waits until it receives a trigger on the PXI star trigger bus before processing the next entry in the scan list.

**Defined Value**

125



# **NISWITCH\_VAL\_REARCONNECTOR**

## **Description**

The switch waits until it receives a trigger on the rear connector.

## Defined Value

1000

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE1**

## **Description**

The switch waits until it receives a trigger on the rear connector module 1.

## **Defined Value**

1021

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE10**

## **Description**

The switch waits until it receives a trigger on the rear connector module 10.

## Defined Value

1030

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE11**

## **Description**

The switch waits until it receives a trigger on the rear connector module 11.

## Defined Value

1031



# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE12**

## **Description**

The switch waits until it receives a trigger on the rear connector module 12.

**Defined Value**

1032

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE2**

## **Description**

The switch waits until it receives a trigger on the rear connector module 2.

## Defined Value

1022

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE3**

## **Description**

The switch waits until it receives a trigger on the rear connector module 3.

## Defined Value

1023

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE4**

## **Description**

The switch waits until it receives a trigger on the rear connector module 4.

## Defined Value

1024



# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE5**

## **Description**

The switch waits until it receives a trigger on the rear connector module 5.

**Defined Value**

1025

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE6**

## **Description**

The switch waits until it receives a trigger on the rear connector module 6.

## Defined Value

1026

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE7**

## **Description**

The switch waits until it receives a trigger on the rear connector module 7.

**Defined Value**

1027

# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE8**

## **Description**

The switch waits until it receives a trigger on the rear connector module 8.

## Defined Value

1028



# **NISWITCH\_VAL\_REARCONNECTOR\_MODULE9**

## **Description**

The switch waits until it receives a trigger on the rear connector module 9.

## **Defined Value**

1029

## **NISWITCH\_VAL\_RISING\_EDGE**

### **Description**

The trigger occurs on the rising edge of the signal.

**Defined Value**

0

## **NISWITCH\_VAL\_SINGLE**

### **Description**

When scanning, the switch does not share trigger lines with other switches. You must set NISWITCH\_ATTR\_SCAN\_ADVANCED\_OUTPUT and NISWITCH\_ATTR\_TRIGGER\_INPUT for this device.

**Defined Value**

0

# **NISWITCH\_VAL\_SLAVE**

## **Description**

Multiple switches are sharing trigger lines for the scan and this device is one of the trigger slaves. You must set NISWITCH\_ATTR\_MASTER\_SLAVE\_TRIGGER\_BUS and NISWITCH\_ATTR\_MASTER\_SLAVE\_SCAN\_ADVANCED\_BUS for this device.

## Defined Value

2



# NISWITCH\_VAL\_SW\_TRIG\_FUNC

## Description

The switch waits until you call the [niSwitch\\_SendSoftwareTrigger](#) function before processing the next entry in the scan list.

## Defined Value

3

## **NISWITCH\_VAL\_TTL0**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG0 line before processing the next entry in the scan list.

## Defined Value

111

## **NISWITCH\_VAL\_TTL1**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG1 line before processing the next entry in the scan list.

## Defined Value

112

## **NISWITCH\_VAL\_TTL2**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG2 line before processing the next entry in the scan list.

## Defined Value

113



## **NISWITCH\_VAL\_TTL3**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG3 line before processing the next entry in the scan list.

## Defined Value

114

## **NISWITCH\_VAL\_TTL4**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG4 line before processing the next entry in the scan list.

**Defined Value**

115

## **NISWITCH\_VAL\_TTL5**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG5 line before processing the next entry in the scan list.

## Defined Value

116

## **NISWITCH\_VAL\_TTL6**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG6 line before processing the next entry in the scan list.

## Defined Value

117



## **NISWITCH\_VAL\_TTL7**

### **Description**

The switch waits until it receives a trigger on the PXI\_TRIG7 line before processing the next entry in the scan list.

**Defined Value**

118

**VI\_FALSE**

**Description**

False.

**Defined Value**

0

**VI\_TRUE**

**Description**

True.

# Defined Value

1

Group/Attribute Name	Attribute I
<b>Channel Configuration</b>	
Is Source Channel	<u>NISWITCH_ATTR_IS_SOURCE_CH</u>
Is Configuration Channel	<u>NISWITCH_ATTR_IS_CONFIGURA</u>
<b>Module Characteristics</b>	
Serial Number	<u>NISWITCH_ATTR_SERIAL_NUMBI</u>
Is Debounced	<u>NISWITCH_ATTR_IS_DEBOUNCE</u>
Settling Time	<u>NISWITCH_ATTR_SETTLING_TIM</u>
Bandwidth	<u>NISWITCH_ATTR_BANDWIDTH</u>
Maximum DC Voltage	<u>NISWITCH_ATTR_MAX_DC_VOLT</u>
Maximum AC Voltage	<u>NISWITCH_ATTR_MAX_AC_VOLT</u>
Maximum Switching DC Current	<u>NISWITCH_ATTR_MAX_SWITCHIN</u>
Maximum Switching AC Current	<u>NISWITCH_ATTR_MAX_SWITCHIN</u>
Maximum Carry DC Current	<u>NISWITCH_ATTR_MAX_CARRY_D</u>
Maximum Carry AC Current	<u>NISWITCH_ATTR_MAX_CARRY_A</u>
Maximum Switching DC Power	<u>NISWITCH_ATTR_MAX_SWITCHIN</u>
Maximum Switching AC Power	<u>NISWITCH_ATTR_MAX_SWITCHIN</u>
Maximum Carry DC Power	<u>NISWITCH_ATTR_MAX_CARRY_D</u>
Maximum Carry AC Power	<u>NISWITCH_ATTR_MAX_CARRY_A</u>
Characteristic Impedance	<u>NISWITCH_ATTR_CHARACTERIS'</u>
Wire mode	<u>NISWITCH_ATTR_WIRE_MODE</u>
Number of Relays	<u>NISWITCH_ATTR_NUMBER_OF_F</u>
<b>Scanning Configuration</b>	
Scan List	<u>NISWITCH_ATTR_SCAN_LIST</u>
Scan Mode	<u>NISWITCH_ATTR_SCAN_MODE</u>
Continuous Scan	<u>NISWITCH_ATTR_CONTINUOUS_</u>
Trigger Input	<u>NISWITCH_ATTR_TRIGGER_INPL</u>
Scan Advanced Output	<u>NISWITCH_ATTR_SCAN_ADVANC</u>
Is Scanning	<u>NISWITCH_ATTR_IS_SCANNING</u>

Is Waiting for Trigger?	<a href="#"><u>NISWITCH_ATTR_IS_WAITING_FC</u></a>
Scan Delay	<a href="#"><u>NISWITCH_ATTR_SCAN_DELAY</u></a>
Trigger Input Polarity	<a href="#"><u>NISWITCH_ATTR_TRIGGER_INPL</u></a>
Scan Advanced Polarity	<a href="#"><u>NISWITCH_ATTR_SCAN_ADVANC</u></a>
Handshaking Initiation	<a href="#"><u>NISWITCH_ATTR_HANDSHAKING</u></a>
<b>Matrix Configuration</b>	
Number of Rows	<a href="#"><u>NISWITCH_ATTR_NUM_OF_ROW</u></a>
Number of Columns	<a href="#"><u>NISWITCH_ATTR_NUM_OF_COLL</u></a>
<b>Obsolete Attributes</b>	
Cabled Module Scan Advanced Bus	<a href="#"><u>NISWITCH_ATTR_CABLED_MODU</u></a>
Cabled Module Trigger Bus	<a href="#"><u>NISWITCH_ATTR_CABLED_MODU</u></a>
Master Slave Scan Advanced Bus	<a href="#"><u>NISWITCH_ATTR_MASTER_SLAV</u></a>
Master Slave Trigger Bus	<a href="#"><u>NISWITCH_ATTR_MASTER_SLAV</u></a>
Parsed Scan List	<a href="#"><u>NISWITCH_ATTR_PARSED_SCAN</u></a>
Trigger Mode	<a href="#"><u>NISWITCH_ATTR_TRIGGER_MOD</u></a>



# NISWITCH\_ATTR\_IS\_CONFIGURATION\_CHANNE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

This channel-based attribute specifies whether to reserve the channel for internal path creation. A channel that is available for internal path creation is called a configuration channel. The driver may use configuration channels to create paths between two channels you specify in the [niSwitch\\_Connect](#) function. Configuration channels are not available for external connections.

Set this attribute to [VI\\_TRUE](#) to mark the channel as a configuration channel. Set this attribute to [VI\\_FALSE](#) to mark the channel as available for external connections.

After you identify a channel as a configuration channel, you cannot use that channel for external connections. The [niSwitch\\_Connect](#) function returns the NISWITCH\_ERROR\_IS\_CONFIGURATION\_CHANNEL error when you attempt to establish a connection between a configuration channel and any other channel.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_IS\_SOURCE\_CHANNEL

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

This channel-based attribute specifies whether you want to identify the channel as a source channel. Typically, you set this attribute to [VI\\_TRUE](#) when you attach the channel to a power supply, a function generator, or an active measurement point on the unit under test, and you do not want to connect the channel to another source. The driver prevents source channels from connecting to each other. The [niSwitch\\_Connect](#) function returns the NISWITCH\_ERROR\_ATTEMPT\_TO\_CONNECT\_SOURCES when you attempt to connect two channels that you identify as source channels.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_DRIVER\_SETUP

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## Description

This attribute indicates the Driver Setup string that the user specified when initializing the driver.

Some cases exist where the end-user must specify instrument driver options at initialization time. An example of this is specifying a particular instrument model from among a family of instruments that the driver supports. This is useful when using simulation. The end-user can specify driver-specific options through the DriverSetup keyword in the optionString parameter to the [niSwitch\\_InitWithOptions](#) function, or through the IVI Configuration Utility.

If the user does not specify a Driver Setup string, this attribute returns an empty string.



# NISWITCH\_ATTR\_IO\_RESOURCE\_DESCRIPTOR

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## **Description**

Indicates the resource descriptor the driver uses to identify the physical device.

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

If you initialize the instrument driver with the resource descriptor, this attribute contains that value.

# NISWITCH\_ATTR\_LOGICAL\_NAME

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## Description

A string containing the logical name you specified when opening the current IVI session.

You may pass a logical name to the [niSwitch\\_init](#) or [niSwitch\\_InitWithOptions](#) functions. The IVI Configuration utility must contain an entry for the logical name. The logical name entry refers to a virtual instrument section in the IVI Configuration file. The virtual instrument section specifies a physical device and initial user options.

# NISWITCH\_ATTR\_CHANNEL\_COUNT

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	RO	N/A	None	None

## **Description**

Indicates the number of channels that the specific instrument driver supports.

# NISWITCH\_ATTR\_GROUP\_CAPABILITIES

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

**Description**

A string that contains a comma-separated list of class-extension groups that this driver implements.



# NISWITCH\_ATTR\_SUPPORTED\_INSTRUMENT\_M

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

**Description**

Contains a comma-separated list of supported instrument models.

# NISWITCH\_ATTR\_SPECIFIC\_DRIVER\_CLASS\_SF

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	RO	N/A	None	None

**Description**

The major version number of the IviSwtch class specification.

# NISWITCH\_ATTR\_SPECIFIC\_DRIVER\_CLASS\_SF

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	RO	N/A	None	None

**Description**

The minor version number of the class specification with which this driver is compliant.

# NISWITCH\_ATTR\_SPECIFIC\_DRIVER\_DESCRIPTOR

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

**Description**

A string that contains a brief description of the specific driver.



# NISWITCH\_ATTR\_SPECIFIC\_DRIVER\_PREFIX

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## **Description**

A string that contains the prefix for the instrument driver. The name of each user-callable function in this driver starts with this prefix.

# NISWITCH\_ATTR\_SPECIFIC\_DRIVER\_VENDOR

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

**Description**

A string that contains the name of the vendor that supplies this driver.

# NISWITCH\_ATTR\_SPECIFIC\_DRIVER\_REVISION

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## **Description**

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

# NISWITCH\_ATTR\_INSTRUMENT\_FIRMWARE\_RE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## **Description**

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



# NISWITCH\_ATTR\_INSTRUMENT\_MANUFACTURE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

**Description**

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

# NISWITCH\_ATTR\_INSTRUMENT\_MODEL

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## **Description**

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

# NISWITCH\_ATTR\_CACHE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

Specifies whether to cache the value of attributes. When caching is enabled, the instrument driver keeps track of the current instrument settings and avoids sending redundant commands to the instrument. The instrument driver can choose always to cache or never to cache particular attributes regardless of the setting of this attribute.

The default value is [VI\\_TRUE](#). Use the [niSwitch\\_InitWithOptions](#) function to override this value.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_INTERCHANGE\_CHECK

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None



## Description

Specifies whether to perform interchangeability checking and retrieve interchangeability warnings when you call [niSwitch\\_Connect](#), [niSwitch\\_SetPath](#) and [niSwitch\\_InitiateScan](#) functions.

The default value is [VI\\_FALSE](#).

Interchangeability warnings indicate that using your application with a different instrument might cause different behavior. call [niSwitch\\_GetNextInterchangeWarning](#) to extract interchange warnings. Call the [niSwitch\\_ClearInterchangeWarnings](#) 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 instrument and you have not set that attribute, or the attribute has been invalidated since you set it.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_QUERY\_INSTRUMENT\_STATUS

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

Specifies whether the instrument driver queries the instrument status after each operation. Querying the instrument status is very useful for debugging. After you validate your program, you can set this attribute to [VI\\_FALSE](#) to disable status checking and maximize performance

The instrument driver can choose to ignore status checking for particular attributes regardless of the setting of this attribute.

The default value is [VI\\_TRUE](#). Use the [niSwitch\\_InitWithOptions](#) function to override this value.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_RANGE\_CHECK

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

Specifies whether to validate attribute values and function parameters. If enabled, the instrument driver validates the parameter values that you pass to driver functions. Range checking parameters is very 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 [niSwitch\\_InitWithOptions](#) function to override this value.

## Defined Values:

VI\_TRUE

VI\_FALSE



# NISWITCH\_ATTR\_RECORD\_COERCIONS

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

Specifies whether the IVI engine keeps a list of the value coercions it makes for ViInt32 and ViReal64 attributes. call [niSwitch\\_GetNextCoercionRecord](#) to extract and delete the oldest coercion record from the list.

The default value is [VI\\_FALSE](#). Use the [niSwitch\\_InitWithOptions](#) function to override this value.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_SIMULATE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

Specifies whether or not to simulate instrument driver I/O operations. If simulation is enabled, instrument driver functions perform range checking and call `Ivi_GetAttribute` and `Ivi_SetAttribute` functions, but they do not perform instrument I/O. For output parameters that represent instrument data, the instrument driver functions return calculated values.

The default value is [VI\\_FALSE](#). Use the [niSwitch\\_InitWithOptions](#) function to override this value.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_NUM\_OF\_COLUMNS

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	RO	N/A	None	None

## Description

This attribute returns the number of channels on the column of a matrix or scanner. If the switch is a scanner, this value is the number of input channels.

The NISWITCH\_ATTR\_WIRE\_MODE attribute affects the number of available columns. For example, if your device has 8 input lines and you use the four-wire mode, then the number of columns you have available is 2.



# NISWITCH\_ATTR\_NUM\_OF\_ROWS

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	RO	N/A	None	None

## Description

This attribute returns the number of channels on the row of a matrix or scanner. If the switch is a scanner, this value is the number of output channels.

The NISWITCH\_ATTR\_WIRE\_MODE attribute affects the number of available rows. For example, if your device has 8 input lines and you use the two-wire mode, then the number of columns you have available is 4.

# NISWITCH\_ATTR\_BANDWIDTH

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the bandwidth for the channel.  
The units are hertz.

# NISWITCH\_ATTR\_CHARACTERISTIC\_IMPEDANC

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the characteristic impedance for the channel.

The units are ohms.

# NISWITCH\_ATTR\_IS\_DEBOUNCED

## Specific Attribute

Data type	Access	Applies to	Coercion	High Level Functions
ViBoolean RO	N/A	None	None	

## Description

This attribute indicates whether the entire switch has settled since the last switching command. A value of VI\_TRUE indicates that all signals going through the switch are valid.



## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_MAX\_AC\_VOLTAGE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum AC voltage the channel can switch.

The units are volts RMS.

# NISWITCH\_ATTR\_MAX\_CARRY\_AC\_CURRENT

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum AC current the channel can carry.

The units are amperes RMS.

# NISWITCH\_ATTR\_MAX\_CARRY\_AC\_POWER

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum AC power the channel can carry.

The units are volt-amperes.

# NISWITCH\_ATTR\_MAX\_CARRY\_DC\_CURRENT

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None



## **Description**

This channel-based attribute returns the maximum DC current the channel can carry.

The units are amperes.

# NISWITCH\_ATTR\_MAX\_CARRY\_DC\_POWER

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum DC power the channel can carry.

The units are watts.

# NISWITCH\_ATTR\_MAX\_DC\_VOLTAGE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum DC voltage the channel can switch.

The units are volts.

# NISWITCH\_ATTR\_MAX\_SWITCHING\_AC\_CURRE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum AC current the channel can switch.

The units are amperes RMS.

# NISWITCH\_ATTR\_MAX\_SWITCHING\_AC\_POWER

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None



## **Description**

This channel-based attribute returns the maximum AC power the channel can switch.

The units are volt-amperes.

# NISWITCH\_ATTR\_MAX\_SWITCHING\_DC\_CURRE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum DC current the channel can switch.

The units are amperes.

# NISWITCH\_ATTR\_MAX\_SWITCHING\_DC\_POWER

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	RO	N/A	None	None

## **Description**

This channel-based attribute returns the maximum DC power the channel can switch.

The units are watts.

# NISWITCH\_ATTR\_NUMBER\_OF\_RELAYS

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	RO	N/A	None	None

## **Description**

This attribute returns the number of relays.

# NISWITCH\_ATTR\_POWER\_DOWN\_LATCHING\_RI

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None



## Description

This attribute indicates whether to power down latching relays after calling Wait For Debounce. When Power Down Latching Relays After Debounce is enabled (VI TRUE), a call to Wait For Debounce ensures that the relays are settled and the latching relays are powered down.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_SERIAL\_NUMBER

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## **Description**

This read-only attribute returns the serial number for the switch controlled by NI-SWITCH. If the device does not return a serial number, NI-SWITCH returns the Invalid Attribute error.

# NISWITCH\_ATTR\_SETTLING\_TIME

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	R/W	N/A	None	None

## Description

This channel-based attribute returns the maximum length of time from after you make a connection until the signal flowing through the channel [settles](#). The units are seconds.



**Note** PXI-2501/2503/2565/2590/2591 Users—the actual delay will always be the greater value of the settling time and the value you specify as the [scan delay](#).

# NISWITCH\_ATTR\_WIRE\_MODE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	RO	N/A	None	None

## Description

This attribute returns the wire mode of the switch.

This attribute affects the values of the

[NISWITCH\\_ATTR\\_NUM\\_OF\\_ROWS](#) and

[NISWITCH\\_ATTR\\_NUM\\_OF\\_COLUMNS](#) attributes. The actual number of input and output lines on the switch is fixed, but the number of channels depends on how many lines constitute each channel.



# NISWITCH\_ATTR\_CABLED\_MODULE\_SCAN\_AD'

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## Description

This attribute has been deprecated and may be removed from a future release of NI-SWITCH. Use the [niSwitch\\_RouteScanAdvancedOutput](#) function instead.

## **Defined Values:**

NISWITCH\_VAL\_NONE

NISWITCH\_VAL\_TTL0

NISWITCH\_VAL\_TTL1

NISWITCH\_VAL\_TTL2

NISWITCH\_VAL\_TTL3

NISWITCH\_VAL\_TTL4

NISWITCH\_VAL\_TTL5

NISWITCH\_VAL\_TTL6

NISWITCH\_VAL\_TTL7

# NISWITCH\_ATTR\_CABLED\_MODULE\_TRIGGER\_

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## Description

This attribute has been deprecated and may be removed from a future release of NI-SWITCH. Use the [niSwitch\\_RouteTriggerInput](#) function instead.

## **Defined Values:**

NISWITCH\_VAL\_NONE

NISWITCH\_VAL\_TTL0

NISWITCH\_VAL\_TTL1

NISWITCH\_VAL\_TTL2

NISWITCH\_VAL\_TTL3

NISWITCH\_VAL\_TTL4

NISWITCH\_VAL\_TTL5

NISWITCH\_VAL\_TTL6

NISWITCH\_VAL\_TTL7

# NISWITCH\_ATTR\_CONTINUOUS\_SCAN

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	R/W	N/A	None	None

## Description

When a switch is scanning, the switch can either stop scanning when the end of the scan (VI\_FALSE) or continue scanning from the top of the scan list again (VI\_TRUE).

Notice that if you set the scan to continuous (VI\_TRUE), the Wait For Scan Complete operation will always time out and you must call Abort to stop the scan.



## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_HANDSHAKING\_INITIATION

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## Description

# NISWITCH\_ATTR\_IS\_SCANNING

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	RO	N/A	None	None

## **Description**

This attribute indicates whether the switch has completed the scan operation. The value VI\_TRUE indicates that the scan is complete.

## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_IS\_WAITING\_FOR\_TRIG

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViBoolean	RO	N/A	None	None

## **Description**

In a scan list, a semicolon (;) is used to indicate that at that point in the scan list, the scan engine should pause until a trigger is received from the trigger input. If that trigger is user generated through either a hardware pulse or the Send SW Trigger operation, it is necessary for the user to know when the scan engine has reached such a state.



## Defined Values:

VI\_TRUE

VI\_FALSE

# NISWITCH\_ATTR\_PARSED\_SCAN\_LIST

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	RO	N/A	None	None

## **Description**

This attribute has been deprecated and may be removed from a future release of NI-SWITCH.

# NISWITCH\_ATTR\_MASTER\_SLAVE\_SCAN\_ADVA

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## **Description**

This attribute has been deprecated and may be removed from a future release of NI-SWITCH. Use the [niSwitch\\_RouteScanAdvancedOutput](#) function instead.

## **Defined Values:**

NISWITCH\_VAL\_NONE

NISWITCH\_VAL\_TTL0

NISWITCH\_VAL\_TTL1

NISWITCH\_VAL\_TTL2

NISWITCH\_VAL\_TTL3

NISWITCH\_VAL\_TTL4

NISWITCH\_VAL\_TTL5

NISWITCH\_VAL\_TTL6

NISWITCH\_VAL\_TTL7

# NISWITCH\_ATTR\_SCAN\_ADVANCED\_OUTPUT

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## **Description**

This attribute specifies the method you want to use to notify another instrument that all signals going through the switch have settled following the processing of one entry in the scan list.



## **Defined Values:**

NISWITCH\_VAL\_NONE

NISWITCH\_VAL\_EXTERNAL

NISWITCH\_VAL\_TTL0

NISWITCH\_VAL\_TTL1

NISWITCH\_VAL\_TTL2

NISWITCH\_VAL\_TTL3

NISWITCH\_VAL\_TTL4

NISWITCH\_VAL\_TTL5

NISWITCH\_VAL\_TTL6

NISWITCH\_VAL\_TTL7

NISWITCH\_VAL\_REARCONNECTOR

NISWITCH\_VAL\_REARCONNECTOR\_MODULE1

NISWITCH\_VAL\_REARCONNECTOR\_MODULE2

NISWITCH\_VAL\_REARCONNECTOR\_MODULE3

NISWITCH\_VAL\_REARCONNECTOR\_MODULE4

NISWITCH\_VAL\_REARCONNECTOR\_MODULE5

NISWITCH\_VAL\_REARCONNECTOR\_MODULE6

NISWITCH\_VAL\_REARCONNECTOR\_MODULE7

NISWITCH\_VAL\_REARCONNECTOR\_MODULE8

NISWITCH\_VAL\_REARCONNECTOR\_MODULE9

NISWITCH\_VAL\_REARCONNECTOR\_MODULE10

NISWITCH\_VAL\_REARCONNECTOR\_MODULE11

NISWITCH\_VAL\_REARCONNECTOR\_MODULE12

NISWITCH\_VAL\_FRONTCONNECTOR

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE1

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE2

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE3

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE4

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE5

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE6

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE7

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE8

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE9

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE10

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE11

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE12



## Notes

- (0) NISWITCH\_VAL\_NONE The switch does not produce a Scan Advanced Output trigger.
- (2) NISWITCH\_VAL\_EXTERNAL External Trigger. The switch produces the Scan Advanced Output trigger on the "trigger out" connector.
- (111) NISWITCH\_VAL\_TTL0 The switch produces the Scan Advanced Output on the SCXI or PXI\_TRIG0 line.
- (112) NISWITCH\_VAL\_TTL1 The switch produces the Scan Advanced Output on the PXI\_TRIG1 line.
- (113) NISWITCH\_VAL\_TTL2 The switch produces the Scan Advanced Output on the SCXI or PXI\_TRIG2 line.
- (114) NISWITCH\_VAL\_TTL3 The switch produces the Scan Advanced Output on the PXI\_TRIG3 line.
- (115) NISWITCH\_VAL\_TTL4 The switch produces the Scan Advanced Output on the PXI\_TRIG4 line.
- (116) NISWITCH\_VAL\_TTL5 The switch produces the Scan

Advanced Output on the PXI\_TRIG5 line.

- (117) NISWITCH\_VAL\_TTL6 The switch produces the Scan Advanced Output on the PXI\_TRIG6 line.
- (118) NISWITCH\_VAL\_TTL7 The switch produces the Scan Advanced Output on the PXI\_TRIG7 line.
- (125) NISWITCH\_VAL\_PXI\_STAR The switch produces the Scan Advanced Output on the PXI STAR trigger bus.
- (1001) NISWITCH\_VAL\_FRONTCONNECTOR This indicates that the switch will send its SCANNER ADVANCED output to the front connector. When using SCXI switches as scanners, all the devices that are part of the scanner will send their SCANNER ADVANCED output to their respective front connectors.

# NISWITCH\_ATTR\_SCAN\_ADVANCED\_POLARITY

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

**Description**

**Defined Values:**

NISWITCH\_VAL\_RISING\_EDGE

NISWITCH\_VAL\_FALLING\_EDGE

# NISWITCH\_ATTR\_SCAN\_DELAY

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViReal64	R/W	N/A	None	None



## Description

This attribute specifies the minimum amount of time the switch waits before it asserts the scan advanced output trigger after opening or closing the switch. The switch always waits for debounce before asserting the trigger. The units are seconds.



**Note** PXI-2501/2503/2565/2590/2591 Users—the actual delay will always be the greater value of the **settling time** and the value you specify as the scan delay.

# NISWITCH\_ATTR\_SCAN\_LIST

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViString	R/W	N/A	None	None

## Description

This attribute contains a [scan list](#)—a string that specifies channel connections and trigger conditions. The [niSwitch\\_InitiateScan](#) function makes or breaks connections and waits for triggers according to the instructions in the scan list. A scan list is comprised of channel names that you separate with special characters. These special characters determine the operations the scanner performs on the channels when it executes this scan list.

- To create a path between two channels, use the following character between the two channel names: -> (a dash followed by a '>' sign) Example: \CH1->CH2\ tells the switch to make a path from channel CH1 to channel CH2.
- To break or clear a path, use the following character as a prefix before the path: ~ (tilde) Example: \~CH1->CH2\ tells the switch to break the path from channel CH1 to channel CH2.
- To tell the switch to wait for a trigger event, use the following character as a separator between paths: ; (semicolon) Example: \CH1->CH2;CH3->CH4\ tells the switch to make the path from channel CH1 to channel CH2, wait for a trigger, and then make the path from CH3 to CH4.
- To tell the switch to create multiple paths simultaneously, use the following character as a separator between the paths: , (comma) Example: \A->B;CH1->CH2,CH3->CH4\ instructs the scanner to make the path between channels A and B, wait for a trigger, and then simultaneously make the paths between channels CH1 and CH2 and between channels CH3 and CH4.

Refer to [Scan Lists](#) for additional information.

# NISWITCH\_ATTR\_SCAN\_MODE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## **Description**

This attribute specifies what happens to existing connections that conflict with the connections you make in a scan list. For example, if CH1 is already connected to CH2 and the scan list instructs the switch to connect CH1 to CH3, this attribute specifies what happens to the connection between CH1 and CH2.

If the value of this attribute is `NISWITCH_VAL_NONE`, the switch takes no action on existing paths. If the value is `NISWITCH_VAL_BREAK_BEFORE_MAKE`, the switch breaks conflicting paths before making new ones. If the value is `NISWITCH_VAL_BREAK_AFTER_MAKE`, the switch breaks conflicting paths after making new ones.

Most switches support only one of the possible values. In such cases, this attribute serves as an indicator of the device's behavior.

**Defined Values:**

NISWITCH\_VAL\_NONE

NISWITCH\_VAL\_BREAK\_BEFORE\_MAKE

NISWITCH\_VAL\_BREAK\_AFTER\_MAKE

# NISWITCH\_ATTR\_MASTER\_SLAVE\_TRIGGER\_BI

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## Description

This attribute has been deprecated and may be removed from a future release of NI-SWITCH. Use the [niSwitch\\_RouteTriggerInput](#) function instead.



## **Defined Values:**

NISWITCH\_VAL\_NONE

NISWITCH\_VAL\_TTL0

NISWITCH\_VAL\_TTL1

NISWITCH\_VAL\_TTL2

NISWITCH\_VAL\_TTL3

NISWITCH\_VAL\_TTL4

NISWITCH\_VAL\_TTL5

NISWITCH\_VAL\_TTL6

NISWITCH\_VAL\_TTL7

NISWITCH\_VAL\_FRONTCONNECTOR

# NISWITCH\_ATTR\_TRIGGER\_INPUT

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## **Description**

This attribute specifies the source of the trigger for which the switch can wait when processing a scan list. The switch waits for a trigger when it encounters a semicolon in a scan list. When the trigger occurs, the switch advances to the next entry in the scan list.

## **Defined Values:**

NISWITCH\_VAL\_IMMEDIATE

NISWITCH\_VAL\_EXTERNAL

NISWITCH\_VAL\_SOFTWARE\_TRIG

NISWITCH\_VAL\_TTL0

NISWITCH\_VAL\_TTL1

NISWITCH\_VAL\_TTL2

NISWITCH\_VAL\_TTL3

NISWITCH\_VAL\_TTL4

NISWITCH\_VAL\_TTL5

NISWITCH\_VAL\_TTL6

NISWITCH\_VAL\_TTL7

NISWITCH\_VAL\_PXI\_STAR

NISWITCH\_VAL\_REARCONNECTOR

NISWITCH\_VAL\_REARCONNECTOR\_MODULE1

NISWITCH\_VAL\_REARCONNECTOR\_MODULE2

NISWITCH\_VAL\_REARCONNECTOR\_MODULE3

NISWITCH\_VAL\_REARCONNECTOR\_MODULE4

NISWITCH\_VAL\_REARCONNECTOR\_MODULE5

NISWITCH\_VAL\_REARCONNECTOR\_MODULE6

NISWITCH\_VAL\_REARCONNECTOR\_MODULE7

NISWITCH\_VAL\_REARCONNECTOR\_MODULE8

NISWITCH\_VAL\_REARCONNECTOR\_MODULE9

NISWITCH\_VAL\_REARCONNECTOR\_MODULE10

NISWITCH\_VAL\_REARCONNECTOR\_MODULE11

NISWITCH\_VAL\_REARCONNECTOR\_MODULE12

NISWITCH\_VAL\_FRONTCONNECTOR

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE1

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE2

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE3

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE4

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE5

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE6

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE7

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE8

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE9

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE10

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE11

NISWITCH\_VAL\_FRONTCONNECTOR\_MODULE12



## Notes

- (1) NISWITCH\_VAL\_IMMEDIATE Immediate Trigger. The switch does not wait for a trigger before processing the next entry in the scan list.
- (2) NISWITCH\_VAL\_EXTERNAL External Trigger. The switch waits until it receives a trigger from an external source through the "trigger in" connector.
- (3) NISWITCH\_VAL\_SOFTWARE\_TRIG The switch waits until you call the [niSwitch\\_SendSWTrigger](#) function.
- (111) NISWITCH\_VAL\_TTL0 The switch waits until it receives a trigger on the SCXI or PXI\_TRIG0 line before processing the next entry in the scan list.

- (112) NISWITCH\_VAL\_TTL1 The switch waits until it receives a trigger on the PXI\_TRIG1 line before processing the next entry in the scan list.
- (113) NISWITCH\_VAL\_TTL2 The switch waits until it receives a trigger on the SCXI or PXI\_TRIG2 line before processing the next entry in the scan list.
- (114) NISWITCH\_VAL\_TTL3 The switch waits until it receives a trigger on the PXI\_TRIG3 line before processing the next entry in the scan list.
- (115) NISWITCH\_VAL\_TTL4 The switch waits until it receives a trigger on the PXI\_TRIG4 line before processing the next entry in the scan list.
- (116) NISWITCH\_VAL\_TTL5 The switch waits until it receives a trigger on the PXI\_TRIG5 line before processing the next entry in the scan list.
- (117) NISWITCH\_VAL\_TTL6 The switch waits until it receives a trigger on the PXI\_TRIG6 line before processing the next entry in the scan list.
- (118) NISWITCH\_VAL\_TTL7 The switch waits until it receives a trigger on the PXI\_TRIG7 line before processing the next entry in the scan list.
- (125) NISWITCH\_VAL\_PXI\_STAR The switch waits until it receives a trigger on the PXI STAR trigger bus before processing the next entry in the scan list.
- (1000) NISWITCH\_VAL\_REARCONNECTOR The switch waits until it receives a trigger on the Rear connector before processing the next entry in the scan list. This value is valid for SCXI scanners that consist of a single device. If more than one device is used, you must use [niSwitch\\_RouteTriggerInput](#) or [niSwitch\\_RouteScanAdvancedOutput](#) functions to route a trigger from the connector on another module to one of the TTL lines instead.
- (1001) NISWITCH\_VAL\_FRONTCONNECTOR The switch waits until it receives a trigger on the front connector before processing the next entry in the scan list. When using SCXI scanners, this variable is valid for scanners that consist of a

single device. If more than one device is used, you must use the [niSwitch\\_RouteTriggerInput](#) or [niSwitch\\_RouteScanAdvancedOutput](#) functions to route a trigger from the connector on another module to one of the TTL lines instead.



# NISWITCH\_ATTR\_TRIGGER\_INPUT\_POLARITY

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

**Description**

**Defined Values:**

NISWITCH\_VAL\_RISING\_EDGE

NISWITCH\_VAL\_FALLING\_EDGE

# NISWITCH\_ATTR\_TRIGGER\_MODE

## Specific Attribute

<b>Data type</b>	<b>Access</b>	<b>Applies to</b>	<b>Coercion</b>	<b>High Level Functions</b>
ViInt32	R/W	N/A	None	None

## Description

This attribute has been deprecated and may be removed from a future release of NI-SWITCH. Use the [niSwitch\\_RouteTriggerInput](#) and/or [niSwitch\\_RouteScanAdvancedOutput](#) functions instead.

## **Defined Values:**

NISWITCH\_VAL\_SINGLE

NISWITCH\_VAL\_MASTER

NISWITCH\_VAL\_SLAVE