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.

<table>
<thead>
<tr>
<th>Class/Panel Name</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td><code>niSwitch_init</code></td>
</tr>
<tr>
<td>Initialize With Options</td>
<td><code>niSwitch_InitWithOptions</code></td>
</tr>
<tr>
<td>Initialize With Topology</td>
<td><code>niSwitch_InitWithTopology</code></td>
</tr>
</tbody>
</table>

### Configuration Functions

#### Set/Get/Check Attribute

#### Set Attribute

- Set Attribute ViInt32: `niSwitch_SetAttributeViInt32`
- Set Attribute ViReal64: `niSwitch_SetAttributeViReal64`
- Set Attribute ViString: `niSwitch_SetAttributeViString`
- Set Attribute ViBoolean: `niSwitch_SetAttributeViBoolean`
- Set Attribute ViSession: `niSwitch_SetAttributeViSession`

#### Get Attribute

- Get Attribute ViInt32: `niSwitch_GetAttributeViInt32`
- Get Attribute ViReal64: `niSwitch_GetAttributeViReal64`
- Get Attribute ViString: `niSwitch_GetAttributeViString`
- Get Attribute ViBoolean: `niSwitch_GetAttributeViBoolean`
- Get Attribute ViSession: `niSwitch_GetAttributeViSession`

#### Check Attribute

- Check Attribute ViInt32: `niSwitch_CheckAttributeViInt32`
- Check Attribute ViReal64: `niSwitch_CheckAttributeViReal64`
- Check Attribute ViString: `niSwitch_CheckAttributeViString`
- Check Attribute ViBoolean: `niSwitch_CheckAttributeViBoolean`
- Check Attribute ViSession: `niSwitch_CheckAttributeViSession`

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

Commit: niSwitch_Commit
Get Channel Name: niSwitch_GetChannelName
Reset: niSwitch_reset
Reset With Defaults: niSwitch_ResetWithDefaults
Disable: niSwitch_Disable
Self-Test: niSwitch_self_test
Revision Query: niSwitch_revision_query
Error-Query: niSwitch_error_query
Error Message: niSwitch_error_message

Coercion Info

Get Next Coercion Record: niSwitch_GetNextCoercionRecord

Interchangeability Info

Get Next Interchange Warning: niSwitch_GetNextInterchangeWarning
Clear Interchange Warnings: niSwitch_ClearInterchangeWarnings
Reset Interchange Check: niSwitch_ResetInterchangeCheck

Error Info

Get Error: niSwitch_GetError
Clear Error: niSwitch_ClearError

Locking

Lock Session: niSwitch_LockSession
Unlock Session: niSwitch_UnlockSession
Close: niSwitch_close
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceName</td>
<td>ViRsrc</td>
<td>Resource name of the switch module to initialize.</td>
</tr>
</tbody>
</table>

Syntax:

<table>
<thead>
<tr>
<th>MAX</th>
<th>Configured under Valid Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI-DAQmx Devices</td>
<td>DAQmxDeviceName</td>
</tr>
<tr>
<td>Traditional NI-DAQ (Legacy) Devices</td>
<td>SCXI[chassis ID]::slot number</td>
</tr>
<tr>
<td>PXI System</td>
<td>PXI[bus number]::device number</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC1Mod3</td>
<td>NI-DAQmx module in chassis &quot;SC1&quot; Slot 3</td>
</tr>
<tr>
<td>MySwitch</td>
<td>NI-DAQmx module renamed to &quot;MySwitch&quot;</td>
</tr>
<tr>
<td>SCXI1::3</td>
<td>Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3</td>
</tr>
<tr>
<td>SCXI::3</td>
<td>Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3</td>
</tr>
<tr>
<td>PXI0::16</td>
<td>PXI bus 0, device number 16</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI_TRUE</td>
<td>Queries the switch to determine which device is installed. (default)</td>
</tr>
<tr>
<td>VI_FALSE</td>
<td>Currently unsupported.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI_TRUE</td>
<td>Resets the device. (default)</td>
</tr>
<tr>
<td>VI_FALSE</td>
<td>Currently unsupported. The device will not reset.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceName</td>
<td>ViRsrc</td>
<td>Resource name of the switch module to initialize.</td>
</tr>
</tbody>
</table>

**Syntax:**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX</td>
<td>Configured under Valid Syntax</td>
</tr>
<tr>
<td>NI-DAQmx Devices</td>
<td>DAQmxDeviceName</td>
</tr>
<tr>
<td>Traditional NI-DAQ (Legacy) Devices</td>
<td>SCXI[chassis ID]:slot number</td>
</tr>
<tr>
<td>PXI System</td>
<td>PXI[business number]:device number</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC1Mod3</td>
<td>NI-DAQmx module in chassis 'SC1', Slot 3</td>
</tr>
<tr>
<td>MySwitch</td>
<td>NI-DAQmx module renamed to 'MySwitch'</td>
</tr>
<tr>
<td>SCXI1::3</td>
<td>Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3</td>
</tr>
<tr>
<td>SCXI::3</td>
<td>Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3</td>
</tr>
<tr>
<td>PXI0::16</td>
<td>PXI bus 0, device number 16</td>
</tr>
<tr>
<td>PXI::16</td>
<td>PXI bus 0, device number 16</td>
</tr>
</tbody>
</table>

| 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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI_TRUE</td>
<td>(default) Queries the switch to</td>
</tr>
<tr>
<td>VI_FALSE</td>
<td>Currently unsupported</td>
</tr>
</tbody>
</table>
**resetDevice**  
ViBoolean  
Specifies whether to reset the switch module during the initialization process.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI_TRUE</td>
<td>(default) Reset device</td>
</tr>
<tr>
<td>VI_FALSE</td>
<td>Currently unsupported. The device will not reset.</td>
</tr>
</tbody>
</table>

**optionString**  
ViConstString  
Sets initial values of certain attributes for the NI-SWITCH session. The following table lists the attribute string names you can use:

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NISWITCH_ATTR_RANGE_CHECK</td>
</tr>
<tr>
<td>NISWITCH_ATTR_QUERY_INSTRUMENT_STATUS</td>
</tr>
<tr>
<td>NISWITCH_ATTR_CACHE</td>
</tr>
<tr>
<td>NISWITCH_ATTR_SIMULATE</td>
</tr>
<tr>
<td>NISWITCH_ATTR_RECORD_COERCIONS</td>
</tr>
<tr>
<td>NISWITCH_ATTR_DRIVER_SETUP</td>
</tr>
</tbody>
</table>

The format of the **optionString** is, "AttributeStringName=Value" where AttributeStringName is the name of the attribute shown above and Value is the value to which the attribute will be set. To set multiple attributes, separate assignments with a comma.

If you pass an empty string for this parameter, the NI-SWITCH session uses the default values for the attributes. You can override the default values by explicitly assigning a value. You do not have to specify all the available attributes. If you do not specify an attribute, its default value is used.

Use the DriverSetup attribute to set the topology or the resource type (DAQmx or Traditional DAQ) of the switch module. This attribute can contain config token/value pairs within it.

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

Valid Config Tokens and Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
topology
The topology of the device.
for valid values.

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

For example, use the following string to set a NI SCXI-1127 as a 2-wire 32x1 multiplexer configured in MAX under DAQmx Devices:

"DriverSetup=topology:1127/2-Wire 32x1 Mux;resourcetype:daqmx"

The DriverSetup string is particularly important when using NI-SWITCH through the IviSwtch class driver.

To enable simulation, set simulate equal to 1 and topology of the switch module to simulate. For example, simulation for an NI SCXI-1127 configured as a 2-wire 32x1 multiplexer:

"Simulate=1, DriverSetup=topology:1127/2-Wire 32x1 Mux"

If simulate is set to 1 and the DriverSetup topology is used to determine which device to simulate, the device specified in resource name is simulated if the DriverSetup string does not specify a topology.

vi ViSession
A particular NI-SWITCH session established with niSwitch_InitWithOptions, or niSwitch_init, is used for all subsequent NI-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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceName</td>
<td>ViRsrc</td>
<td>Resource name of the switch module to initialize.</td>
</tr>
</tbody>
</table>

Syntax:

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

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC1Mod3</td>
<td>NI-DAQmx module in chassis &quot;SC1&quot; Slot 3</td>
</tr>
<tr>
<td>MySwitch</td>
<td>NI-DAQmx module renamed to &quot;MySwitch&quot;</td>
</tr>
<tr>
<td>SCXI1::3</td>
<td>Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3</td>
</tr>
<tr>
<td>SCXI::3</td>
<td>Traditional NI-DAQ (Legacy) module in chassis 1, Slot 3</td>
</tr>
<tr>
<td>PXI0::16</td>
<td>PXI bus 0, device number 16</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI_TRUE</td>
<td>Simulate</td>
</tr>
<tr>
<td>VI_FALSE</td>
<td>(default) Do not simulate</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI_TRUE</td>
<td>Reset device</td>
</tr>
<tr>
<td>(default)</td>
<td>The device will not reset.</td>
</tr>
<tr>
<td>VI_FALSE</td>
<td></td>
</tr>
</tbody>
</table>

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`. 
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithOptions</code>, <code>niSwitch_InitWithTopology</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vi</strong></td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td><strong>channelName</strong></td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td><strong>attributeID</strong></td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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 is 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <em>niSwitch_InitWithTopology</em>, <em>niSwitch_InitWithOptions</em>, or <em>niSwitch_init</em> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vi</strong></td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with niSwitch_InitWithTopology, niSwitch_InitWithOptions, or niSwitch_init and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td><strong>channelName</strong></td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td><strong>attributeID</strong></td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with [niSwitch_InitWithTopology], [niSwitch_InitWithOptions], or [niSwitch_init] and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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 NULL byte, contains more bytes that 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 that 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| vi           | ViSession     | A particular NI-SWITCH session established with
niSwitch_InitWithTopology,
iSwitch_InitWithOptions, or niSwitch_init
and used for all subsequent NI-SWITCH calls.                                                                                     |
| channelName  | 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. |
| attributeID  | 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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 ViInt32 attribute.
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>Pass the ID of an attribute.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| vi         | ViSession    | A particular NI-SWITCH session established with
              *niSwitch_InitWithTopology*,
              *niSwitch_InitWithOptions*, or *niSwitch_init*
              and used for all subsequent NI-SWITCH calls. |
| **channelName** | 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. |
| **attributeID** | 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 <ctl-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. |
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with \texttt{niSwitch_InitWithTopology}, \texttt{niSwitch_InitWithOptions}, or \texttt{niSwitch_init} and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with \texttt{niSwitch_InitWithTopology}, \texttt{niSwitch_InitWithOptions}, or \texttt{niSwitch_init} and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channelName</td>
<td>ViConstString</td>
<td>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.</td>
</tr>
<tr>
<td>attributeID</td>
<td>ViAttr</td>
<td>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 &lt;ENTER&gt;, &lt;spacebar&gt;, or &lt;ctrl-down arrow&gt;, 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 &lt;ENTER&gt;. 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</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>channel1</td>
<td>ViConstString</td>
<td>Input one of the channel names of the desired path. Pass the other channel name as <code>channel2</code>. Refer to <code>Devices</code> for valid channel names for the switch module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples of valid channel names:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ch0, com0, ab0, r1, c2, cjtemp</td>
</tr>
<tr>
<td>channel2</td>
<td>ViConstString</td>
<td>Input one of the channel names of the desired path. Pass the other channel name as <code>channel1</code>. Refer to <code>Devices</code> for valid channel names for the switch module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples of valid channel names:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ch0, com0, ab0, r1, c2, cjtemp</td>
</tr>
</tbody>
</table>
niSwitch_Disconnect

IviSwtchBase 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 \texttt{niSwitch\_Connect} or \texttt{niSwitch\_SetPath} function.
### Parameters

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

IviSwtchBase 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
niSwitch_IsDebounced

IviSwtchBase 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>isDebounced</td>
<td>ViBoolean</td>
<td><strong>VI_TRUE</strong> indicates that all created paths have settled. <strong>VI_FALSE</strong> indicates that all created paths have not settled.</td>
</tr>
</tbody>
</table>
niSwitch_WaitForDebounce

IviSwtchBase 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>maximumTime_ms</td>
<td>ViInt32</td>
<td>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.</td>
</tr>
</tbody>
</table>
niSwitch_CanConnect
IviSwitchBase 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vi</strong></td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td><strong>channel1</strong></td>
<td>ViConstString</td>
<td>Input one of the channel names of the desired path. Pass the other channel name as the <strong>channel2</strong>. Refer to 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.</td>
</tr>
<tr>
<td><strong>channel2</strong></td>
<td>ViConstString</td>
<td>Input one of the channel names of the desired path. Pass the other channel name as <strong>channel1</strong>. Refer to channel names for the switch module. Examples of valid channel names: ch0, com0, ab0, r1, c2, cjtemp The default value is an empty string.</td>
</tr>
<tr>
<td><strong>pathCapability</strong></td>
<td>ViInt32</td>
<td>Indicates whether a path is valid. Possible values include: <code>NISWITCH_VAL_PATH_AVAILABLE</code>, <code>NISWITCH_VAL_PATH_EXISTS</code></td>
</tr>
</tbody>
</table>
NISWITCH_VAL_PATH_UNSUPPORTED

NISWITCH_VAL_RSRC_IN_USE

NISWITCH_VAL_SOURCE_CONFLICT
NISWITCH_VAL_CHANNEL_NOT_AVAIL
niSwitch_SetPath

IviSwtchBase 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>pathList</td>
<td>ViConstString</td>
<td>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-&gt;com0, com0-&gt;ab0. In this example, com0 is a configuration channel. Obtain the path list for a previously created path with <code>niSwitch_GetPath</code>.</td>
</tr>
</tbody>
</table>
niSwitch_GetPath

IviSwtchBase 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 \texttt{niSwitch\_Connect}. Pass this string to \texttt{niSwitch\_SetPath} to establish the exact same path in future connections.

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

\texttt{niSwitch\_GetPath} only returns those paths explicitly created by \texttt{niSwitch\_Connect} or \texttt{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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vi</strong></td>
<td>ViSession</td>
<td>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.</td>
</tr>
<tr>
<td><strong>channel1</strong></td>
<td>ViConstString</td>
<td>Input one of the channel names of the desired path. Pass the other channel name as <strong>channel2</strong>. 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.</td>
</tr>
<tr>
<td><strong>channel2</strong></td>
<td>ViConstString</td>
<td>Input one of the channel names of the desired path. Pass the other channel name as <strong>channel1</strong>. 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.</td>
</tr>
<tr>
<td><strong>bufferSize</strong></td>
<td>ViInt32</td>
<td>Pass the number of bytes in the ViChar array you specify for the <strong>Path</strong> parameter. If the current value of the attribute, including the terminating NULL byte, contains more bytes that you indicate in this parameter, the function copies <strong>bufferSize–1</strong> 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 &quot;R1-&gt;C1&quot; and <strong>bufferSize</strong> is 4, the function places &quot;R1-&quot; into the buffer and returns 7. If you pass 0, you...</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code> used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>scanlist</td>
<td>ViConstString</td>
<td>Pass the scan list you want the instrument to use.</td>
</tr>
<tr>
<td>initiation</td>
<td>ViInt16</td>
<td>Use the <strong>initiation</strong> parameter to specify whether the measurement device initiates the scan trigger handshake. This parameter determines whether to wait for the scan to reach a trigger point before completing. If the measurement device initiates the scan, set <code>NISWITCH_VAL_MEASUREMENT_DEVICE_INITIATED</code> then waits until the switch is waiting for a trigger from the measurement device before completing. If the switch initiates the scan, set this parameter to <code>NISWITCH_VAL_SWITCH_INITIATED</code>. This function then completes immediately after initiating the scan. You should have already set up your DMM to wait for a trigger before calling this function with <strong>Initiation</strong> set to <code>NISWITCH_VAL_SWITCH_INITIATED</code>.</td>
</tr>
</tbody>
</table>

**Value**

- `NISWITCH_VAL_SWITCH_INITIATED`
- `NISWITCH_VAL_MEASUREMENT_DEVICE_INITIATED` (default)
niSwitch_InitiateScan
IviSwitchScanner 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
niSwitch_AbortScan

IviSwitchScanner 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>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.</td>
</tr>
</tbody>
</table>
niSwitch_SendSoftwareTrigger

IviSwtchSoftwareTrigger 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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
niSwitch_IsScanning

IviSwtchScanner Capability Group

C Function Prototype

ViStatus niSwitch_IsScanning (ViSession vi, ViBoolean* isScanning);
Purpose
Indicates the status of the scan.
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>isScanning</td>
<td>ViBoolean</td>
<td>NI-SWITCH returns the value of <code>NISWITCH_ATTR_IS_SCANNING</code> attribute. <code>VI_TRUE</code> indicates that the switch is scanning. <code>VI_FALSE</code> indicates that the switch is idle.</td>
</tr>
</tbody>
</table>
niSwitch_WaitForScanComplete

IviSwtchScanner 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with \texttt{niSwitch_InitWithTopology}, \texttt{niSwitch_InitWithOptions}, or \texttt{niSwitch_init} is used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>maximumTime_ms</td>
<td>ViInt32</td>
<td>Specifies the maximum length of time to wait for the switch module to stop scanning. If the specified time elapses before the scan ends, the NISWITCH_ERROR_MAX_TIME_EXCEEDED error is returned. The default value is 5000 ms.</td>
</tr>
</tbody>
</table>
niSwitch_SetContinuousScan

IviSwitchScanner 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vi</strong></td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td><strong>continuousScan</strong></td>
<td>ViBoolean</td>
<td>If <strong>VI_TRUE</strong>, loops continuously through the scan list during scanning. If <strong>VI_FALSE</strong>, the scan stops after one pass through the scan list. The default value is <strong>VI_FALSE</strong>.</td>
</tr>
</tbody>
</table>
niSwitch_ConfigureScanList

IviSwtchScanner 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>scanlist</td>
<td>ViConstString</td>
<td>The scan list to use. NI-SWITCH uses this value to set the NISWITCH_ATTR_SCAN_LIST attribute.</td>
</tr>
<tr>
<td>scanMode</td>
<td>ViInt32</td>
<td>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.</td>
</tr>
</tbody>
</table>
niSwitch_ConfigureScanTrigger

IviSwitchScanner 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <strong>niSwitch_InitWithTopology</strong>, <strong>niSwitch_InitWithOptions</strong>, or <strong>niSwitch_init</strong> used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>scanDelay</td>
<td>ViReal64</td>
<td>The minimum length of time you want to wait after it creates a path until it asserts a trigger on the scan advanced output line. The driver uses this value to set the <strong>NISWITCH_ATTR_SCAN_DELAY</strong> attribute. The scan delay is in addition to the settling time. The driver uses this value to set the <strong>NISWITCH_ATTR_SCAN_DELAY</strong> attribute. Express this value in seconds. The default value is 0.0 s.</td>
</tr>
<tr>
<td>triggerInput</td>
<td>ViInt32</td>
<td>Trigger source you want the switch module to use during scanning. The driver uses this value to set the <strong>NISWITCH_ATTR_TRIGGER_INPUT</strong> attribute. The switch waits for the trigger you specify when it encounters a semicolon in the scan list. When the trigger occurs, the switch advances to the next entry in the scan list. This parameter specifies the <strong>NISWITCH_ATTR_TRIGGER_INPUT</strong> of valid values.</td>
</tr>
<tr>
<td>scanAdvancedOutput</td>
<td>ViInt32</td>
<td>Output destination of the scan advanced output signal. NI-SWITCH uses this value to set the <strong>NISWITCH_ATTR_SCAN_ADVANCED_OUTPUT</strong> attribute. After the switch processes every entry in the scan list, it waits the length of time you specify in the <strong>scanDelay</strong> parameter and then asserts a trigger on the line you specify with this parameter. Refer to...</td>
</tr>
</tbody>
</table>
NISWITCH_ATTR_SCAN_ADVANCED
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>triggerInputConnector</td>
<td>ViInt32</td>
<td>The location of the input trigger source on the switch module. Valid locations are the <code>NISWITCH_VAL_FRONTCONNECTOR</code> and <code>NISWITCH_VAL_REARCONNECTOR</code>. The default value is <code>NISWITCH_VAL_FRONTCONNECTOR</code>.</td>
</tr>
<tr>
<td>triggerInputBusLine</td>
<td>ViInt32</td>
<td>The trigger line to route the input trigger. Select <code>NISWITCH_VAL_NONE</code> to break an existing route.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid Values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_NONE</code> (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL0</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL1</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL2</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL3</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL4</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL5</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL6</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>NISWITCH_VAL_TTL7</code></td>
</tr>
<tr>
<td>invert</td>
<td>ViBoolean</td>
<td>If <code>VI_TRUE</code>, inverts the input trigger signal from falling to rising or vice versa. The default value is <code>VI_FALSE</code>.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>

#### `scanAdvancedOutputConnector` ViInt32

The scan advanced output trigger destination.

Valid Values:
- `NISWITCH_VAL_FRONTCONNECTOR` (default)
- `NISWITCH_VAL_REARCONNECTOR`

#### `scanAdvancedOutputBusLine` ViInt32

The trigger line to route the scan advanced output trigger from the front or rear connector. Select `NISWITCH_VAL_NONE` to break an existing route.

Valid Values:
- `NISWITCH_VAL_NONE` (default)
- `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 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vi</strong></td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td><strong>index</strong></td>
<td>ViInt32</td>
<td>A 1-based index into the channel table. The default value is 1. The maximum value is the value of the <code>NISWITCH_ATTR_CHANNEL_COUNT</code> attribute.</td>
</tr>
<tr>
<td><strong>relayNameBufferSize</strong></td>
<td>ViInt32</td>
<td>Pass the number of bytes in the ViChar array you specify for the <code>relayNameBuffer</code> 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 &quot;123456&quot; and <code>relayBufferSize</code> is 4, the function places &quot;123&quot; 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 <code>VI_NULL</code> for the Coercion Record buffer parameter.</td>
</tr>
<tr>
<td>relayNameBuffer</td>
<td>ViChar[]</td>
<td>Returns the relay name for the index you specify.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>relayName</td>
<td>ViConstString</td>
<td>Name of the relay. Refer to <a href="#">Devices</a> for a list of valid relay names for the switch module.</td>
</tr>
<tr>
<td>relayCount</td>
<td>ViInt32</td>
<td>The number of relay cycles.</td>
</tr>
</tbody>
</table>

Examples of valid relay names:

- ch0, ab0, 1wire, hlselect
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>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.</td>
</tr>
<tr>
<td>relayName</td>
<td>ViConstString</td>
<td>Name of the relay. Refer to <a href="#">Devices</a> for a list of valid relay names for the switch module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ch0, ab0, 1wire, hlselect</td>
</tr>
<tr>
<td>relayPosition</td>
<td>ViInt32</td>
<td>Indicates whether the relay is open or closed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="#">NISWITCH_VAL_OPEN</a> (10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="#">NISWITCH_VAL_CLOSED</a> (11)</td>
</tr>
</tbody>
</table>

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

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

Examples of valid relay names:

- ch0, ab0, 1wire, hlselect

<table>
<thead>
<tr>
<th>relayAction</th>
<th>ViInt32</th>
<th>Specifies whether to open or close a given relay.</th>
</tr>
</thead>
</table>

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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with \niSwitch_InitWithTopology, \niSwitch_InitWithOptions, or \niSwitch_init and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with niSwitch_InitWithTopology, niSwitch_InitWithOptions, or niSwitch_init and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>index</td>
<td>ViInt32</td>
<td>A 1-based index into the channel table. The default value is 1. The maximum value is Value of Channel Count attribute.</td>
</tr>
<tr>
<td>bufferSize</td>
<td>ViInt32</td>
<td>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 &quot;123456&quot; and bufferSize is 4, the function places &quot;123&quot; 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.</td>
</tr>
<tr>
<td>channelNameBuffer</td>
<td>ViChar[]</td>
<td>Returns the channel name that is in</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with &lt;code&gt;niSwitch_InitWithTopology&lt;/code&gt;, &lt;code&gt;niSwitch_InitWithOptions&lt;/code&gt;, or &lt;code&gt;niSwitch_init&lt;/code&gt; and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>selfTestResult</td>
<td>ViInt16</td>
<td>Value returned from the switch self-test.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 Passed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Failed</td>
</tr>
<tr>
<td>selfTestMessage</td>
<td>ViChar[]</td>
<td>Self-test response string from the switch. You must pass a ViChar array with at least 256 bytes.</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>instrumentDriverRevision</td>
<td>ViChar[]</td>
<td>NI-SWITCH software revision numbers in the form of a string. You must pass a ViChar array with at least 256 bytes.</td>
</tr>
<tr>
<td>firmwareRevision</td>
<td>ViChar[]</td>
<td>Currently unsupported.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>errorCode</td>
<td>ViInt32</td>
<td>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.</td>
</tr>
<tr>
<td>errorMessage</td>
<td>ViChar[]</td>
<td>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 No error.</td>
</tr>
</tbody>
</table>
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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>bufferSize</td>
<td>ViInt32</td>
<td>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 <code>bufferSize–1</code> 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 &quot;123456&quot; and the <code>bufferSize</code> is 4, the function places &quot;123&quot; 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 <code>VI_NULL</code> for the Coercion Record buffer parameter.</td>
</tr>
<tr>
<td>coercionRecord</td>
<td>ViChar[]</td>
<td>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 <code>bufferSize</code> parameter. If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate with the <code>bufferSize</code> parameter, the function copies <code>bufferSize–1</code> 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 &quot;123456&quot; and the <code>bufferSize</code> is 4, the function places &quot;123&quot; 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 <code>VI_NULL</code> for the Coercion Record buffer parameter.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>bufferSize</td>
<td>ViInt32</td>
<td>Pass the number of bytes in the ViChar array you specify for the <code>interchangeWarning</code> parameter. If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies <code>bufferSize–1</code> 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 &quot;123456&quot; and the <code>bufferSize</code> is 4, the function places &quot;123&quot; 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 <code>VI_NULL</code> for the Interchange Warning buffer parameter.</td>
</tr>
<tr>
<td>interchangeWarning</td>
<td>ViChar[]</td>
<td>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</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td>code</td>
<td>ViStatus</td>
<td>Returns the error code for the session or execution thread. If you pass 0 for <code>bufferSize</code>, you can pass <code>VI_NULL</code> for this parameter.</td>
</tr>
<tr>
<td>bufferSize</td>
<td>Vilnt32</td>
<td>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 <code>bufferSize</code>–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 &quot;123456&quot; and <code>bufferSize</code> is 4, the function places &quot;123&quot; 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 <code>VI_NULL</code> for the <code>description</code> buffer parameter.</td>
</tr>
<tr>
<td>description</td>
<td>ViChar[]</td>
<td>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 <code>bufferSize</code> parameter. If the error</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>

**callerHasLock** ViBoolean  
This parameter serves as a convenience. If you do not want to use this parameter, pass `VI_NULL`.  
Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to `VI_FALSE`. Pass the address of the same local variable to any other calls you make to `niSwitch_LockSession` or `niSwitch_UnlockSession` in the same function. The parameter is an input/output parameter. `niSwitch_LockSession` and `niSwitch_UnlockSession` each inspect the current value and take the following actions:

- If the value is `VI_TRUE`, `niSwitch_LockSession` does not lock the session again. If the value is `VI_FALSE`, `niSwitch_LockSession` obtains the lock and sets the value of the parameter to `VI_TRUE`.

- If the value is `VI_FALSE`, `niSwitch_UnlockSession` does not attempt to unlock the session. If the value is `VI_TRUE`, `niSwitch_UnlockSession` releases the lock and sets the value of the parameter to `VI_FALSE`. 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_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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> and used for all subsequent NI-SWITCH calls.</td>
</tr>
</tbody>
</table>

**callerHasLock** ViBoolean This parameter serves as a convenience. If you do not want to use this parameter, pass `VI_NULL`. Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to `VI_FALSE`. Pass the address of the same local variable to any other calls you make to `niSwitch_LockSession` or `niSwitch_UnlockSession` in the same function. The parameter is an input/output parameter. `niSwitch_LockSession` and `niSwitch_UnlockSession` each inspect the current value and take the following actions:

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

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));
  }
  else
  {
    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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| vi           | ViSession  | A particular NI-SWITCH session established with `niSwitch_InitWithTopology`, `niSwitch_InitWithOptions`, or `niSwitch_init` for all subsequent
               |            | NI-SWITCH calls.                                                                                                                          |
| channelName  | 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 channel names.                                                                                 |
|              |            | While `niSwitch_ReadCalibrationData` and `niSwitch_WriteCalibrationData` take a channel name, some switches only support a single calibration for all input channels. |
|              |            | NI PXI-2501 uses an optional single amplifier for channels ch0 through ch47 to decrease settling time. In these cases, writing a different channel causes the previous value to be overwritten. Therefore, reading different channels returns the same calibration data. |
| calibrationField | ViInt32  | Tells NI-SWITCH which particular calibration parameter associated with this channel to read. Valid values depend on the switch hardware.   |
|              |            | Examples of possible values: `NISWITCH_VAL_CALIBRATION_CJS_AMP`, `NISWITCH_VAL_CALIBRATION_CHANNEL_AMP` (1)
<table>
<thead>
<tr>
<th><strong>calibrationData</strong></th>
<th>ViReal64</th>
<th>Calibration data from the EEPROM.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>calibrationDate_Year</strong></td>
<td>ViInt32</td>
<td>Year the switch was last calibrated with this calibration data. For example, the date August 1, 2003 would be returned as 2003 by this parameter.</td>
</tr>
<tr>
<td><strong>calibrationDate_Month</strong></td>
<td>ViInt32</td>
<td>Month the switch was last calibrated with this calibration data. For example, the date August 1, 2003 would be returned as 8 by this parameter.</td>
</tr>
<tr>
<td><strong>calibrationDate_Day</strong></td>
<td>ViInt32</td>
<td>Day the switch was last calibrated with this calibration data. For example, the date August 1, 2003 would be returned as 1 by this parameter.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vi</code></td>
<td>ViSession</td>
<td>A particular NI-SWITCH session established with <code>niSwitch_InitWithTopology</code>, <code>niSwitch_InitWithOptions</code>, or <code>niSwitch_init</code> for all subsequent NI-SWITCH calls.</td>
</tr>
<tr>
<td><code>channelName</code></td>
<td>ViConstString</td>
<td>Name of the channel calibrated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples of valid channel names:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ch0, com0, ab0, r1, c2, cjtemp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to <a href="#">Devices</a> for a complete list of valid channel names.</td>
</tr>
<tr>
<td><code>calibrationField</code></td>
<td>ViInt32</td>
<td>Tells NI-SWITCH which particular calibration parameter associated with this channel to write. Valid values depend on the switch hardware.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples of possible values: <code>NISWITCH_VAL_CALIBRATION_CJS_AMP</code>, <code>NISWITCH_VAL_CALIBRATION_CHANNEL_AMP</code> (1)</td>
</tr>
<tr>
<td><strong>calibrationData</strong></td>
<td>ViReal64</td>
<td>Calibration data to store in the EEPROM</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Group/Attribute Name</th>
<th>Attribute I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channel Configuration</strong></td>
<td></td>
</tr>
<tr>
<td>Is Source Channel</td>
<td>NISWITCH_ATTR_IS_SOURCE_CHANNEL</td>
</tr>
<tr>
<td>Is Configuration Channel</td>
<td>NISWITCH_ATTR_IS_CONFIGURATION_CHANNEL</td>
</tr>
<tr>
<td><strong>Module Characteristics</strong></td>
<td></td>
</tr>
<tr>
<td>Serial Number</td>
<td>NISWITCH_ATTR_SERIAL_NUMBER</td>
</tr>
<tr>
<td>Is Debounced</td>
<td>NISWITCH_ATTR_IS_DEBOUNCED</td>
</tr>
<tr>
<td>Settling Time</td>
<td>NISWITCH_ATTR_SETTLING_TIME</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>NISWITCH_ATTR_BANDWIDTH</td>
</tr>
<tr>
<td>Maximum DC Voltage</td>
<td>NISWITCH_ATTR_MAX_DC_VOLT</td>
</tr>
<tr>
<td>Maximum AC Voltage</td>
<td>NISWITCH_ATTR_MAX_AC_VOLT</td>
</tr>
<tr>
<td>Maximum Switching DC Current</td>
<td>NISWITCH_ATTR_MAX_SWITCHING_DC_CURRENT</td>
</tr>
<tr>
<td>Maximum Switching AC Current</td>
<td>NISWITCH_ATTR_MAX_SWITCHING_AC_CURRENT</td>
</tr>
<tr>
<td>Maximum Carry DC Current</td>
<td>NISWITCH_ATTR_MAX_CARRY_DC</td>
</tr>
<tr>
<td>Maximum Carry AC Current</td>
<td>NISWITCH_ATTR_MAX_CARRY_AC</td>
</tr>
<tr>
<td>Maximum Switching DC Power</td>
<td>NISWITCH_ATTR_MAX_SWITCHING_DC_POWER</td>
</tr>
<tr>
<td>Maximum Switching AC Power</td>
<td>NISWITCH_ATTR_MAX_SWITCHING_AC_POWER</td>
</tr>
<tr>
<td>Maximum Carry DC Power</td>
<td>NISWITCH_ATTR_MAX_CARRY_DC</td>
</tr>
<tr>
<td>Maximum Carry AC Power</td>
<td>NISWITCH_ATTR_MAX_CARRY_AC</td>
</tr>
<tr>
<td>Characteristic Impedance</td>
<td>NISWITCH_ATTR_CHARACTERISTIC_IMPEDANCE</td>
</tr>
<tr>
<td>Wire mode</td>
<td>NISWITCH_ATTR_WIRE_MODE</td>
</tr>
<tr>
<td>Number of Relays</td>
<td>NISWITCH_ATTR_NUMBER_OF_RELAYS</td>
</tr>
<tr>
<td><strong>Scanning Configuration</strong></td>
<td></td>
</tr>
<tr>
<td>Scan List</td>
<td>NISWITCH_ATTR_SCAN_LIST</td>
</tr>
<tr>
<td>Scan Mode</td>
<td>NISWITCH_ATTR_SCAN_MODE</td>
</tr>
<tr>
<td>Continuous Scan</td>
<td>NISWITCH_ATTR_CONTINUOUS</td>
</tr>
<tr>
<td>Trigger Input</td>
<td>NISWITCH_ATTR_TRIGGER_INPUT</td>
</tr>
<tr>
<td>Scan Advanced Output</td>
<td>NISWITCH_ATTR_SCAN_ADVANCED_OUTPUT</td>
</tr>
<tr>
<td>Is Scanning</td>
<td>NISWITCH_ATTR_IS_SCANNING</td>
</tr>
</tbody>
</table>
Is Waiting for Trigger?  NISWITCH_ATTR_IS_WAITING_FC
Scan Delay  NISWITCH_ATTR_SCAN_DELAY
Trigger Input Polarity  NISWITCH_ATTR_TRIGGER_INPUT_POLARITY
Scan Advanced Polarity  NISWITCH_ATTR_SCAN_ADVANCED_POLARITY
Handshaking Initiation  NISWITCH_ATTR_HANDSHAKING

Matrix Configuration
Number of Rows  NISWITCH_ATTR_NUM_OF_ROWS
Number of Columns  NISWITCH_ATTR_NUM_OF_COLUMNS

Obsolete Attributes
Cabled Module Scan Advanced Bus  NISWITCH_ATTR_CABLED_MODULE_SCAN_ADVANCED_BUS
Cabled Module Trigger Bus  NISWITCH_ATTR_CABLED_MODULE_TRIGGER_BUS
Master Slave Scan Advanced Bus  NISWITCH_ATTR_MASTER_SLAVE_SCAN_ADVANCED_BUS
Master Slave Trigger Bus  NISWITCH_ATTR_MASTER_SLAVE_TRIGGER_BUS
Parsed Scan List  NISWITCH_ATTR_PARSED_SCAN
Trigger Mode  NISWITCH_ATTR_TRIGGER_MODE
### NISWITCH_ATTR_IS_CONFIGURATION_CHANNEL

#### Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**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_IORESOURCE_DESCRIPTOR

### Specific Attribute

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Specific Attribute</th>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
Indicates the number of channels that the specific instrument driver supports.
## NISWITCH_ATTR_GROUP_CAPABILITIES

### Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description

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

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
Contains a comma-separated list of supported instrument models.
<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
The major version number of the IviSwtch class specification.
NISWITCH_ATTR_SPECIFIC_DRIVER_CLASS_SF

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
The minor version number of the class specification with which this driver is compliant.
## Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
A string that contains a brief description of the specific driver.
### NISWITCH_ATTR_SPECIFIC_DRIVER_PREFIX

Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description

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

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
A string that contains additional version information about this instrument driver.
### NISWITCH_ATTR_INSTRUMENT_FIRMWARE_REVISION

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**Description**

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

Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
A string that contains the name of the instrument manufacturer you are currently using.
### NISWITCH_ATTR_INSTRUMENT_MODEL

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**Description**

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

Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIInt32</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
This channel-based attribute returns the bandwidth for the channel. The units are hertz.
### NISWITCH_ATTR_CHARACTERISTIC_IMPEDANCE

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description

This channel-based attribute returns the characteristic impedance for the channel.
The units are ohms.
### NISWITCH_ATTR_IS_DEBOUNCED

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReral64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
This channel-based attribute returns the maximum DC voltage the channel can switch.
The units are volts.
### NISWITCH_ATTR_MAX_SWITCHING_AC_CURRENT

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description

This channel-based attribute returns the maximum AC current the channel can switch.
The units are amperes RMS.
<table>
<thead>
<tr>
<th>NISWITCH_ATTR_MAX_SWITCHING_AC_POWER</th>
<th>Specific Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Access</td>
</tr>
<tr>
<td>ViReal64</td>
<td>RO</td>
</tr>
</tbody>
</table>
**Description**

This channel-based attribute returns the maximum AC power the channel can switch.
The units are volt-amperes.
## NISWITCH_ATTR_MAX_SWITCHING_DC_CURRENT

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
This channel-based attribute returns the maximum DC current the channel can switch.
The units are amperes.
<table>
<thead>
<tr>
<th>Specific Attribute</th>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NISWITCH_ATTR_MAX_SWITCHING_DC_POWER</td>
<td>ViReal64</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
This attribute returns the number of relays.
**NISWITCH_ATTR_POWER_DOWN_LATCHING_RELAYS_AFTER_DEBOUNCE**

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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_ADVANCED_BUS

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
### NISWITCH_ATTR_IS_SCANNING

Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
This attribute indicates whether the switch has completed the scan operation. The value \texttt{VI\_TRUE} indicates that the scan is complete.
Defined Values:

angered

VI_TRUE

VI_FALSE
NISWITCH_ATTR_IS_WAITING_FOR_TRIG

Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViBoolean</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**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_PARSEDSCANLIST

#### Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>RO</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
This attribute has been deprecated and may be removed from a future release of NI-SWITCH.
<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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
(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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Defined Values:

NISWITCH_VAL_RISING_EDGE

NISWITCH_VAL_FALLING_EDGE
**NISWITCH_ATTR_SCAN_DELAY**

Specific Attribute

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViReal64</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
**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

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViString</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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_BUS

**Specific Attribute**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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_VALSOFTWARE_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**

<table>
<thead>
<tr>
<th>Data type</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Description
Defined Values:

NISWITCH_VAL_RISING_EDGE

NISWITCH_VAL_FALLING_EDGE
### NISWITCH_ATTR_TRIGGER_MODE

**Specific Attribute**

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Access</th>
<th>Applies to</th>
<th>Coercion</th>
<th>High Level Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViInt32</td>
<td>R/W</td>
<td>N/A</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
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