
NI Switch Executive

2024-09-02



Contents

NI Switch Executive Help.....	14
Fundamentals.....	14
Virtual Device	14
IVI Switch	14
Channel	15
Reserved for Routing.....	15
Exclusion	17
Exclusions and SPDT Relays.....	18
Route.....	18
Route Group	18
Hardwire.....	20
Hardwires and Reserved for Routing Channels.....	20
Bus.....	21
Getting Started	22
Activating Your Software.....	22
Verifying System Requirements	25
NI Switch Executive and IVI	25
Naming Conventions.....	26
Using NI Switch Executive in MAX.....	27
Creating a Virtual Device.....	27
Creating a Virtual Device with NI SwitchBlock.....	29
Creating a Virtual Device Using the Switch SFP.....	33
Configuring a Virtual Device.....	33
Setting Configuration Options.....	34
Configuring IVI Switches.....	35
Adding IVI Switches	36
Removing IVI Switches	36
Using Third-Party Switches	37
Schematic Configuration	38
Getting Started with Schematic Configuration	41
Route Groups.....	42
Adding a Route Group	42

Copying a Route Group	43
Nesting a Route Group	43
Disassociating a Nested Route Group	44
Deleting a Route Group	45
Routes	45
Adding a Route	45
Editing a Route	47
Sharing a Route	47
Disassociating a Route	48
Deleting a Route	49
Configuring a Channel	50
Hardwires	50
Adding a Hardwire	51
Deleting a Hardwire	51
Extended Configuration Tasks	52
Third-Party Switches	52
Independent Topologies	53
Extended Configuration	56
Batch Renaming	56
Sorting, Filtering, and Comments	57
Channels/Exclusions Tab	58
Channels	58
Configuring Channels	59
Exclusions	60
Adding Mutual Exclusions	60
Adding Set Exclusions	61
Deleting Exclusions	61
Hardwires/Buses Tab	62
Hardwires	62
Adding Hardwires to Define Multidevice Topologies ...	62
Deleting Hardwires	63
Buses	63
Adding Buses	63
Deleting Buses	65
Routes/Groups Tab	65

Routes	66
Adding Routes	66
Adding Routes with the Path Editor	69
Deleting Routes	70
Route Groups	70
Adding Route Groups	70
Deleting Route Groups	71
Validating a Virtual Device	71
Generating a Report	72
Exporting a Virtual Device	72
Importing a Virtual Device	74
Deploying a Virtual Device	76
Exporting Configurations	76
Importing Configurations	77
Validating Deployment	78
Deleting a Virtual Device	78
NI Switch Executive Test Panel	79
Accessing the Test Panel	79
Using the Test Panel	80
NI Switch Executive Debug Panel	81
Accessing the Debug Panel	82
Creating a New Debug Panel Window	82
Debug Panel State and Action Log Windows	83
Customizing the Debug Panel	84
Debug Panel Toolbar	84
Debug Panel Keyboard Shortcuts	85
Concepts	85
Configuration	86
Validation	86
Programming	86
Export	87
Import	87
Deployment	87
Report Generation	88
Using the Configuration API	88
Configuration VIs	90

COM Functions	91
Classes, Properties, and Methods.....	92
Data Types for NI Switch Executive Configuration Utility	92
niseCfg_Import.....	92
niseCfg_Export	93
niseCfg_ImportSpecific.....	93
niseCfg_ExportPrevious	94
niseCfg_ExportSpecific.....	95
Bus.....	96
Properties.....	96
ChannelsOnBus	96
Comment	97
Hardwires.....	97
HardwireStartIndex	97
Name.....	98
VirtualDevice	98
Buses.....	99
Properties.....	99
Count.....	99
Item	100
VirtualDevice	101
Methods	101
Add.....	101
Clear	102
ConvertToHardwires	102
Remove.....	103
BusChannelPair	103
Properties.....	103
StartChannel	104
EndChannel	104
Methods	104
SetChannelRange.....	104
Channel	105
Properties.....	105
Comment	106
Enabled.....	106

FormattedName	107
Hardwire	107
IviDevice	108
Name	108
ReservedForRouting	109
SignalCharacteristics	109
VirtualDevice	110
Channels	110
Properties	111
Count	111
Item	111
VirtualDevice	112
Methods	112
Add	112
Clear	113
Remove	113
ChannelsOnBus	114
Properties	114
Count	114
Item	115
Methods	115
Add	115
Clear	116
Remove	117
Exclusion	117
Properties	117
Comment	118
Enabled	118
Mutual	118
Name	119
Set1	119
Set2	120
Type	120
VirtualDevice	121
Exclusions	121
Properties	122

Item	122
Count.....	123
VirtualDevice	123
Methods	123
Add.....	123
Clear	124
Remove.....	125
Hardwire.....	125
Properties.....	126
Channels.....	126
Comment	126
Name.....	127
ParentBus.....	127
VirtualDevice	127
Hardwires.....	128
Properties.....	128
Count.....	128
Item	129
VirtualDevice	130
Methods	130
Add.....	130
Clear	131
Remove.....	131
IviDevice.....	131
Properties.....	132
Channels.....	132
Comment	132
Name.....	133
Virtual Device.....	133
IviDevice2.....	133
Properties.....	134
Channels.....	134
Comment	134
DriverName	135
Name.....	135
TopologyName	136

VirtualDevice	136
IviDevices	136
Properties.....	137
Count (Read Only)	137
Item (Read Only).....	137
VirtualDevice (Read Only).....	138
Methods	138
Add.....	139
Clear	139
Remove.....	140
IviDevices2.....	140
Properties.....	141
Count (Read Only)	141
Item (Read Only).....	141
VirtualDevice (Read Only).....	142
Methods	142
Add.....	142
Clear	143
Remove.....	143
AddEx	144
Route.....	145
Properties.....	145
ChannelsInUse	145
Comment	146
Constraints.....	146
Endpoint1.....	146
Endpoint2.....	147
MulticonnectMode	147
Name.....	148
ParentRouteGroups	148
Specification	149
SpecificationType.....	150
VirtualDevice	150
Routes.....	151
Properties.....	151
Count.....	151

Item	152
VirtualDevice	152
Methods	153
Add.....	153
Clear	154
Remove.....	154
RouteGroup.....	154
Properties.....	155
ChildRoutes	155
ChildRouteGroups	155
Comment	156
Name.....	156
VirtualDevice	156
RouteGroups	157
Properties.....	157
Count.....	157
Item	158
VirtualDevice	159
Methods	159
Add.....	159
Clear	160
Remove.....	160
SignalCharacteristics.....	161
Properties.....	161
Bandwidth	162
FormattedValue	162
Impedance.....	163
MaxAcCarryCurrent	163
MaxDcCarryCurrent	164
MaxAcCarryPower	164
MaxDcCarryPower	165
MaxAcSwitchingCurrent	165
MaxDcSwitchingCurrent.....	165
MaxAcSwitchingPower	166
MaxDcSwitchingPower	166
MaxAcVoltage	167

MaxDcVoltage	167
SettlingTime	167
WireMode	168
VirtualDevice	168
Properties	169
Buses	169
Channels	169
Comment	170
Exclusions	170
Hardwires	170
IviDevices	171
Name	171
RouteGroups	172
Routes	172
Version	172
VirtualDevices	173
Properties	173
Count (Read Only)	173
Item (Read Only)	174
Methods	174
Add	175
Save	175
Remove	176
Using the Configuration API in Visual C++	176
Using the Configuration API in Excel	178
Managing a Virtual Device in Excel	178
Exporting to Excel	179
Excel Workbooks	181
Tab-Delimited Text Files	182
Formatting Configuration Data	182
Formatting Configuration Data: Virtual Device	183
Formatting Configuration Data: IVI Devices	184
Formatting Configuration Data: Channels	185
Formatting Configuration Data: Routes	186
Formatting Configuration Data: Route Groups	188
Formatting Configuration Data: Route Group Children	188

Formatting Configuration Data: Hardwires	189
Formatting Configuration Data: Exclusions	189
Formatting Configuration Data: Mutual Exclusion Channels	190
Formatting Configuration Data: Set Exclusion Set1 Channels	190
Formatting Configuration Data: Set Exclusion Set2 Channels	190
Formatting Configuration Data: Buses	191
Formatting Configuration Data: Bus Channels	191
Adding Custom Data	192
Performing Common Management Tasks	192
Importing Excel Data	196
Programming with NI Switch Executive	196
Getting Started	197
Using NI Switch Executive in LabVIEW	197
Using NI Switch Executive in LabWindows/CVI	198
Using NI Switch Executive in Visual C++	198
Using NI Switch Executive in Visual Basic	199
Using NI Switch Executive in NI TestStand	200
Programming Flow	203
Opening a Session	203
Controlling the Virtual Device	203
Error Handling	204
Closing a Session	204
Route Specification Strings	204
Session Reference Counting	206
Specification Type	207
Multiconnect Mode	207
Using Properties and Attributes	208
Setting Properties and Attributes Before Reading Them	209
Debugging with NI I/O Trace	210
NI Switch Executive LabVIEW Reference	211
NI Switch Executive VIs	211
niSE Open Session	213
niSE Close Session	215
niSE Connect	216
niSE Connect And Disconnect	219
niSE Disconnect	223

niSE Disconnect All	225
niSE Expand Route Spec.....	227
niSE Find Route.....	229
niSE Get All Connections	232
niSE Is Connected.....	233
niSE Is Debounced	235
niSE Wait For Debounce	236
Configuration	238
niseCfg Open Configuration.....	240
niseCfg Save.....	241
niseCfg Close Configuration.....	243
niseCfg Get Object	244
niseCfg Get Objects	257
niseCfg Close Object	282
niseCfg Add Object.....	284
niseCfg Remove Object.....	298
niseCfg Import.....	317
niseCfg Export.....	318
niseCfg Import Specific.....	320
niseCfg Export Previous	321
niseCfg Export Specific	324
Property Nodes.....	325
Low Level	325
IVI.....	326
niSE Get IVI Device Session.....	326
niSE Release IVI Device Session.....	328
NI-SWITCH	329
niSE Get NI-SWITCH Session	330
niSE Release NI-SWITCH Session	332
Using the Standard Functionality for error in Parameters	333
Using the Standard Functionality for error out Parameters.....	334
NI Switch Executive C Function Reference	335
niSE_ClearError	335
niSE_CloseSession.....	336
niSE_Connect	337
niSE_ConnectAndDisconnect	338

niSE_Disconnect.....	342
niSE_DisconnectAll	343
niSE_ExpandRouteSpec	343
niSE_FindRoute	345
niSE_GetAllConnections.....	348
niSE_GetError	349
niSE_GetIviDeviceSession	350
niSE_IsConnected	351
niSE_IsDebounced.....	352
niSE_OpenSession.....	353
niSE_WaitForDebounce	354
NI Switch Executive Visual Basic Function Reference	355
niSE_ClearError	356
niSE_CloseSession.....	356
niSE_Connect	357
niSE_ConnectandDisconnect	359
niSE_Disconnect.....	362
niSE_DisconnectAll	363
niSE_ExpandRouteSpec	364
niSE_FindRoute	366
niSE_GetAllConnections.....	368
niSE_GetError	370
niSE_GetIviDeviceSession	371
niSE_IsConnected	372
niSE_IsDebounced.....	373
niSE_OpenSession.....	374
niSE_WaitForDebounce	375
Error Codes	376
Examples	379
Frequently Asked Questions	380

August 2015, 370404J-01

This help file contains development and programming support for NI Switch Executive. In addition to instructions for creating and configuring an NI Switch Executive virtual device, this help file contains [getting started steps](#) for programming your virtual device using LabVIEW, LabWindows™/CVI™, NI TestStand™, and Microsoft Visual Basic and includes [LabVIEW](#), [C](#), [TestStand](#), and [Visual Basic](#) programming references.

Fundamentals

Expand this topic to view information about the terminology related to NI Switch Executive.

Virtual Device

An NI Switch Executive **virtual device** represents a switching system. As such, a virtual device is composed of the switch modules, channels, hardwires, buses, exclusions, routes, and route group definitions and configurations that are a part of the switching system.

Related Topics

[Creating a Virtual Device \(in MAX\)](#) [Creating a Virtual Device Using the Switch SFP Using the Configuration API](#)

IVI Switch

An **IVI switch** is a switch module that uses an IVI-compliant instrument driver. The IVI Foundation defines IVI-compliance in the **IviSwtch Class Specification**, available at www.ivifoundation.org. IVI-compliant switches include all National Instrument switches and some third-party switches.

Related Topics

[Configuring IVI Switches \(in MAX\)](#) [Using the Configuration API](#)

Channel

A **channel** represents a connection point on a switch module. A channel can have multiple wires and can be associated with a hardware. A relay name is not necessarily a channel.

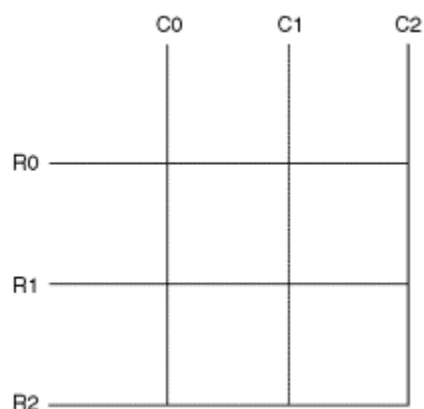
Related Topics

[Schematic Configuration \(in MAX\)](#) [Extended Configuration \(in MAX\)](#) [Using the Configuration API](#)

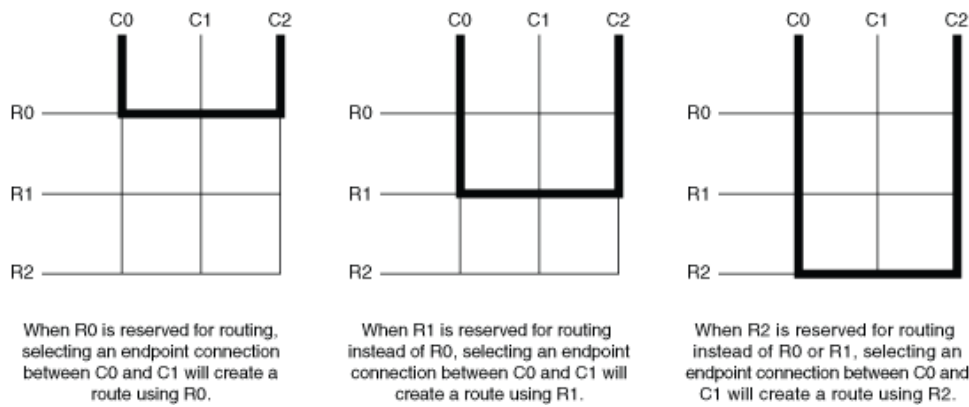
Reserved for Routing

NI Switch Executive uses channels that are **reserved for routing** to complete routes within a switch.

For example, in the following matrix, you want to connect C0 to C2. Depending on the route configurations, you can mark R0, R1, or R2 as reserved for routing to complete the route path. Channels that are reserved for routing cannot be used as endpoints for new routes.



Note Channels that are reserved for routing were called **configuration channels** in earlier versions of NI Switch Executive.



Note Any channel, other than an endpoint, within a [route specification string](#) **must** be marked as reserved for routing or connected using a hardwire to one of the endpoint channels.

Although marking a channel as reserved for routing allows NI Switch Executive to use the channel for completing a route, you still have ultimate control over whether a channel is to be used in a given route.



Note

- In a route, endpoint channels cannot be marked as reserved for routing.
- IVI has a similar concept for configuration channels. Whereas NI Switch Executive honors any configuration channels that are set up as part of the IVI configuration, these configuration channels do not have the same flexibility as an NI Switch Executive channel marked as reserved for routing. [Specific restrictions](#) placed on route specification strings exist for IVI configuration channels.

When configuring routes, if NI Switch Executive cannot find a potential path between channels, you may be recommended to mark certain channels as reserved for routing to complete the route. Choose different channels to use when completing a route by doing one of the following:

- Editing the channel name in the device box or the route string to choose a channel explicitly
- Changing the hardwire selector on a route to choose a channel implicitly



Note When you place routes into route groups, NI Switch Executive automatically routes channels to avoid connection conflicts.

Exclusion

An **exclusion** is a connection rule used to prevent specific channels from connecting to one another. Exclusions are predominately used for safety purposes.



Caution For switch devices with single-pole double-throw (SPDT)—also known as Form C—relays, a physical connection always exists between the common (COM) and normally open (NO) or normally closed (NC) channel. The IVI switch driver may not reflect that connection in all cases, and hazardous or undesired connections can be created. Refer to [Exclusions and SPDT Relays](#) for more information.

Exclusions are a software protection mechanism used to avoid connecting two signals together that should not be connected. For example, if you have power supplies driving two channels on a switch, you can create an exclusion for the channels to help prevent accidental connection of the two signals.



Caution Connecting one signal input to another signal input can create a potentially hazardous and destructive situation.

NI Switch Executive currently offers two types of exclusions—mutual exclusions and set exclusions.

- For [mutual exclusions](#), you create a list of channels that should never connect to one another. A source channel is a type of mutual exclusion.
- For [set exclusions](#), you create two lists of channels. Channels from the first list should never connect to channels of the second list. For example, set exclusions can be used to separate high-power DC and RF signals.

Earlier versions of NI Switch Executive contained a single type of exclusion called the **source channel**. Source channels are a type of mutual exclusion. NI Switch Executive automatically creates a mutual exclusion called `Source Channels` and migrates any source channels from previous versions into this exclusion.

Related Topics

[Extended Configuration \(in MAX\) Using the Configuration API](#)

Exclusions and SPDT Relays

For switch devices with SPDT—also known as Form C—relays, a physical connection always exists between common (COM) and either the normally open (NO) or the normally closed (NC) pole.

When using NI switches, calling [niSE Disconnect](#) with an SPDT relay results in a disconnect in software, but the relay position does not change in hardware until [niSE Connect](#) is called.

Therefore, to guarantee the state of an SPDT relay, you must connect either the NO or the NC pole to COM using [niSE Connect](#). NI Switch Executive verifies that there are no exclusion conflicts before connecting a relay, but does **not** verify exclusions when disconnecting relays using [niSE Disconnect](#) or [niSE DisconnectAll](#).

[niSE DisconnectAll](#) physically resets all relays to their default positions. Because NI Switch Executive does not verify exclusions on disconnect, excluded connections in default relay positions could pose problems for SPDT relays.

Route

A **route** is a connection from one channel (row, column, COM, NO, and so on) to another. Routes are the primary building blocks of any NI Switch Executive virtual device.



Note Some characters are [reserved](#) and cannot be used in route names.

Related Topics

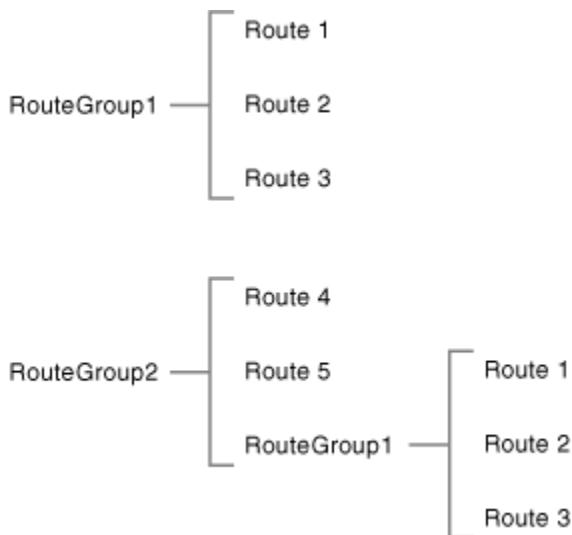
[Schematic Configuration \(in MAX\)](#) [Extended Configuration \(in MAX\)](#) [Using the Configuration API](#)

Route Group

A **route group** is a group of individual routes and/or other route groups associated with a single alias name. Use route groups to set up simultaneous switching

configurations quickly. Route groups also enable NI Switch Executive to make better decisions about which resources to use when creating a route. **When two or more routes are added to a route group, NI Switch Executive ensures these routes do not use conflicting resources.**

For example, the route group named `RouteGroup1`, shown in the following figure, consists of routes `Route1`, `Route2`, and `Route3`. The route group named `RouteGroup2` consists of routes `Route4`, `Route5`, and the route group `RouteGroup1`. When `Connect` calls `RouteGroup1`, all routes within that route group are activated.



In a single test step, multiple switch routes may need to be connected. By grouping those routes, a single call to the [niSE Connect VI](#) or the [niSE_Connect](#) function can connect everything needed for the test step without having to connect each route separately.

With the exception of the way they are created, route and route group names are compatible and are programmatically interchanged. For example, function calls and arguments that expect a route can also use a route group. Routes and route groups can also be used in the same connection/disconnection [specification string](#).



Note Some characters are [reserved](#) and cannot be used in route group names.

Related Topics

[Schematic Configuration \(in MAX\)](#) [Extended Configuration \(in MAX\)](#) [Using the](#)

[Configuration API](#)

Hardwire

A **hardwire** is any physical connection between switch channels, such as wiring, analog buses, matrix expansion plugs, and so on. A hardwire can have any number of channels attached to it.



Note Some characters are [reserved](#) and cannot be used in hardwire names.

Related Topics

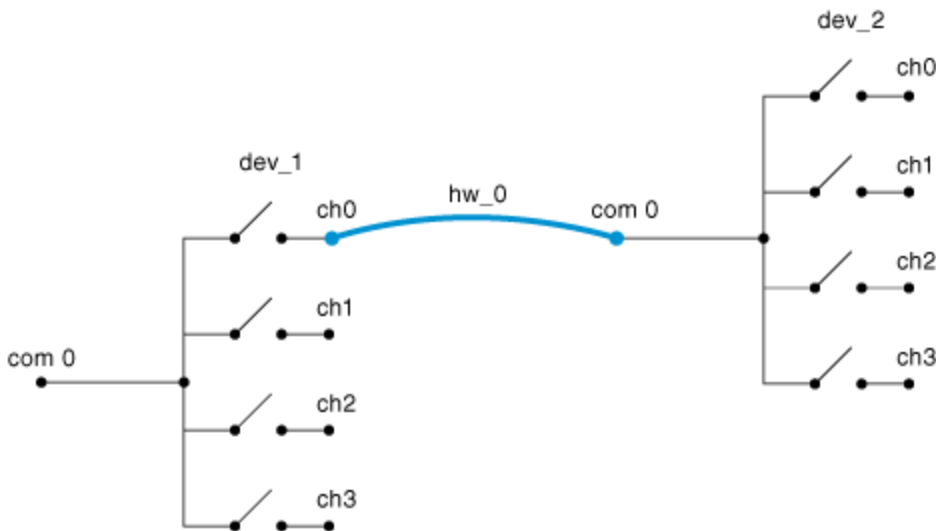
[Schematic Configuration \(in MAX\)](#) [Extended Configuration \(in MAX\)](#) [Using the Configuration API](#)

Hardwires and Reserved for Routing Channels

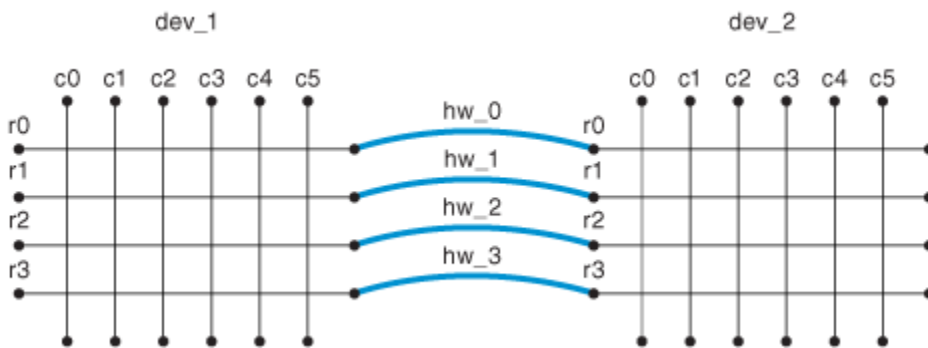
[Reserved for routing](#) is an extension of the IVI configuration channel as defined by the IVI specification and requires intermediate channels on a route to be marked as configuration channels.

NI Switch Executive takes full advantage of the IVI configuration channel definition. By marking channels as reserved for routing, creating a route across multiple devices becomes very easy.

In the following figure, to connect `dev_1/com0` to `dev_2/ch0`, the signal travels across hardwire `hw_0`, `dev_1/ch0`, and `dev_2/com0`. To complete the path, `dev_1/ch0` and `dev_2/com0` should be marked as reserved for routing. Because the IVI configuration channel definition is applied to `dev_1` and `dev_2` as a single device, the two appropriate channels can be marked as reserved for routing.



To simplify marking channels as reserved for routing, NI Switch Executive allows a channel not marked as reserved for routing to be used as an intermediate channel if—and only if—the channel is connected to a hardware connected to an endpoint of the route. For example, in the following figure, `dev_1/r0` can be connected to `dev_2/c0` even though `dev_2/r0` is not marked as reserved for routing.



Tip If you are creating multiple sequential hardwires to a plug or analog bus connector, consider using a [bus](#).

Bus

A **bus** is a group of hardwires associated with a single alias name. Buses are particularly helpful when hardwires are used for column or row expansion in a matrix where a series of channels are connected to another series of channels.



Note Hardwires cannot belong to more than one bus. Hardwires are **not** required to belong

to a bus.

Related Topics

[Schematic Configuration \(in MAX\)](#) [Extended Configuration \(in MAX\)](#) [Using the Configuration API](#)

Getting Started

NI Switch Executive is a configuration application you can use to simplify your switching systems. With NI Switch Executive, you can perform any of the following tasks:

- Develop multiple device switch systems represented as a single virtual device
- Create end-to-end signal routing
- Make route selections based on signal characteristics and switch capabilities
- Create route groups used to make intelligent resource selections when creating routes
- Create switch routing for end-to-end calibration and maximum execution speed
- Create aliases for channels
- Preconfigure routes and route groups that can be called by name at run time
- Control NI and IVI-compliant third-party switches
- Generate a report of your switching system
- Create routes graphically with an interactive schematic view of your switch devices
- Configure large and complex switching systems quickly with Excel integration
- View the state of switches with the debug panel



Note NI Switch Executive currently does **not** support scanning. During scanning you can use triggers to cycle through a set of switching states that have been downloaded to the switch module. You can still perform scanning by [accessing the switch IVI session](#) or by managing the switch outside of NI Switch Executive.

Activating Your Software

How do I activate my software?

Use the NI Activation Wizard to obtain an activation code for your software. You can

launch the NI Activation Wizard two ways:

- Launch the product and choose to activate your software from the list of options presented.
- Launch NI License Manager by selecting **Start»All Programs»National Instruments»NI License Manager** or from NI Launcher in Windows 8. Click the **Activate** button in the toolbar.

Notes



- If your software is a part of a Volume License Agreement (VLA), contact your VLA administrator for installation and activation instructions.
- NI software for OS X and Linux operating systems does not require activation.

What is activation?

Activation is the process of obtaining an activation code to enable your software to run on your computer. An **activation code** is an alphanumeric string that verifies the software, version, and computer ID to enable features on your computer. Activation codes are unique and are valid on only one computer.

What is the NI Activation Wizard?

The NI Activation Wizard is a part of NI License Manager that steps you through the process of enabling software to run on your machine.

What information do I need to activate?

You need your product serial number, user name, and organization. The NI Activation Wizard determines the rest of the information. Certain activation methods may require additional information for delivery. This information is used only to activate your product. Complete disclosure of the National Instruments software licensing information privacy policy is available at ni.com/activate/privacy. If you optionally choose to register your software, your information is protected under the National Instruments privacy policy, available at ni.com/privacy.

How do I find my product serial number?

Your serial number uniquely identifies your purchase of NI software. You can find your serial number on the Certificate of Ownership included in your software kit. If your software kit does not include a Certificate of Ownership, you can find your serial number on the product packing slip or on the shipping label.

If you have installed a previous version using your serial number, you can find the serial number by selecting the **Help»About** menu item within the application or by selecting your product within NI License Manager (**Start»All Programs»National Instruments»NI License Manager** or from NI Launcher in Windows 8). You can also contact your local National Instruments branch at ni.com/contact.

What is a Computer ID?

The computer ID contains unique information about your computer. National Instruments requires this information to enable your software. You can find your computer ID through the NI Activation Wizard or by using NI License Manager, as follows:

1. Launch NI License Manager by selecting **Start»All Programs»National Instruments»NI License Manager** or from NI Launcher in Windows 8.
2. Click the **Display Computer Information** button in the toolbar.

For more information about product activation and licensing refer to ni.com/activate.

How can I evaluate NI software?

You can install and run most NI application software in evaluation mode. This mode lets you use a product with certain limitations, such as reduced functionality or limited execution time. Refer to your product documentation for specific information on the product's evaluation mode.

Moving Software After Activation

To transfer your software to another computer, install and activate it on the second computer. You are not prohibited from transferring your software from one computer to another and you do not need to contact or inform NI of the transfer. Because

activation codes are unique to each computer, you will need a new activation code. Refer to the [How do I activate my software?](#) section of this topic to learn how to acquire a new activation code and reactivate your software.

Deactivating a Product

To deactivate a product and return the product to the state it was in before you activated it, right-click the product in the NI License Manager tree and select **Deactivate**. If the product was in evaluation mode before you activated it, the properties of the evaluation mode may not be restored.

Using Windows Guest Accounts

NI License Manager does not support Microsoft Windows Guest accounts. You must log in to a non-Guest account to run licensed NI application software.

Verifying System Requirements

Verify your system includes the following before you configure an NI Switch Executive virtual device:

- An NI Switch Executive development [license](#)
- The **Development System Requirements** of the [NI Switch Executive Readme](#) file
- An application development environment (ADE), such as LabVIEW, NI TestStand, LabWindows/CVI, or Visual Basic
- [IVI switches](#) with installed IVI-compliant specific instrument drivers

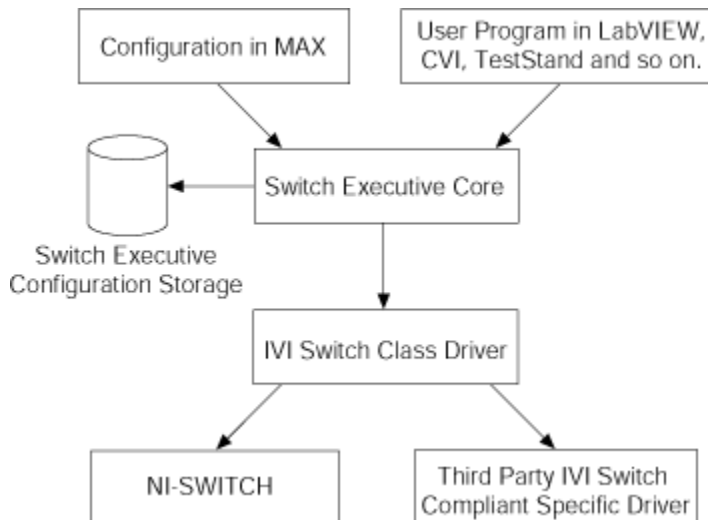


Note NI Switch Executive only supports C-based specific drivers. You can create your own C-based specific IVI driver with LabWindows/CVI templates. For more information about LabWindows/CVI, visit ni.com/cvi.

NI Switch Executive and IVI

NI Switch Executive is built on top of the IVI switch class driver and works with any IVI-compliant switch driver.

The following figure represents the relationship between configuration utilities; application development environments (ADEs) such as LabVIEW, NI TestStand, LabWindows/CVI, or Visual Basic; NI Switch Executive; the IVI switch-class driver; and IVI switch-compliant specific drivers such as NI-SWITCH:



Naming Conventions

Although some NI Switch Executive objects can share the same name, the following are exceptions:

- Routes and route groups cannot share the same name in the same NI Switch Executive virtual device.
- Hardwires and channel aliases cannot share the same name in the same NI Switch Executive virtual device.
- Bus names and hardware start indices cannot result in generated hardware names that conflict with other hardware names or channel aliases.

For example, you cannot name both a route and a route group `Route1` in the same NI Switch Executive virtual device. However, you can name a channel and route group `Power` in the same NI Switch Executive virtual device.

Reserved Characters

The following characters are reserved by NI Switch Executive and must **not** be used in channel, exclusion, hardware, bus, route, route group, or virtual device names. Refer to

Route Specification Strings to learn how some of these characters are used.

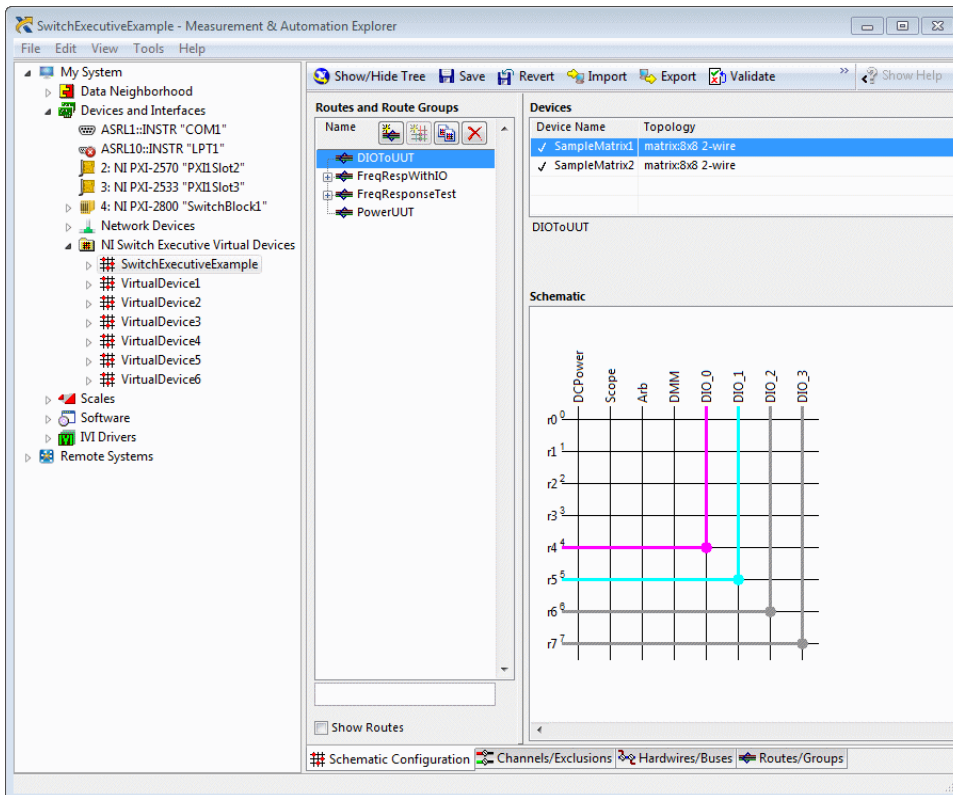
;	,	&
:	!	~
.	< >	[]
{space}	{tab}	/



Note NI Switch Executive ignores white space in route specification strings.

Using NI Switch Executive in MAX

Expand this topic to view instructions about how to use NI Switch Executive in Measurement & Automation Explorer (MAX).




Creating a Virtual Device

Complete the following steps to create an NI Switch Executive virtual device:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces**.
3. Right-click **NI Switch Executive Virtual Devices** and select **Create New**. The Create New dialog box opens.
4. Type a name in the **Virtual Device Name** textbox or accept the predefined Virtual Device Name. The name you select is the name used to refer to the virtual device in the API.



Note Some characters are [reserved](#) and cannot be used in virtual device names.

5. Select a switch device or multiple switch devices from the **Available Switches** listbox and click  to add each switch to the **Switches to Add** listbox.



Note The list of names displayed in the **Available Switches** listbox comes from NI-DAQmx configurations for NI-SWITCH devices and IVI configurations for third-party switches. NI-SWITCH devices automatically display in the **Available Switches** listbox. Refer to [Using Third-Party Switches](#) to create a driver session and logical name for third-party switches to display in the **Available Switches** listbox.

If you cannot locate a device, check your NI-DAQmx and IVI configurations in MAX. If you are using NI switches, you must configure the hardware in MAX as described in the ***NI Switches Getting Started Guide***.

6. (Optional) Click **Configure** if you are using NI switches to configure the settings of those switches.
7. (Optional) Click **Auto Create IVI Devices** to create default IVI configurations for all NI switches detected in your system.
8. Click **Next**. A dialog box displays the status of the virtual device creation.
9. Click **Finish** when the virtual device creation is complete. MAX automatically browses to the NI Switch Executive virtual device that you created.



Note If any errors occur during this process (such as an IVI device not being accessible), the virtual device is still created, but the IVI switches that caused the errors are not added. You can [manually add](#) them later.

Refer to [Adding IVI Switches](#) to continue configuring your NI Switch Executive virtual device.

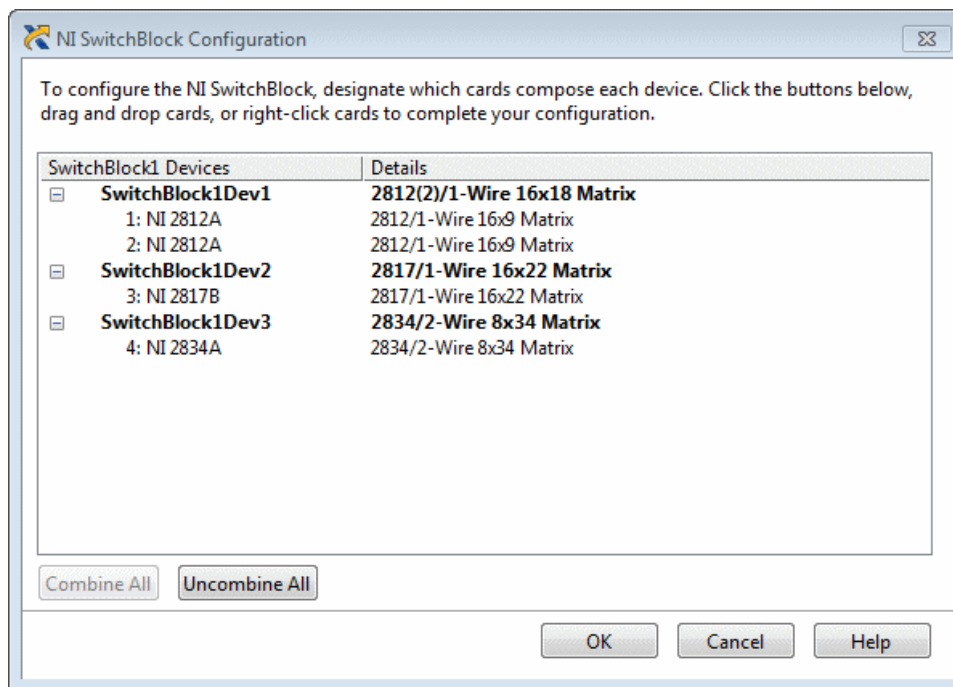
Creating a Virtual Device with NI SwitchBlock

Complete the following steps to create an NI Switch Executive virtual device using NI SwitchBlock cards and carriers:

1. Combine the Cards in Each Carrier

When creating multicarrier virtual devices, combine the cards in each carrier as follows:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces**.
3. Right-click the NI SwitchBlock carrier and select **Configure**. The NI SwitchBlock Configuration dialog box displays the cards in your carrier.
4. Click **Combine All** or drag and drop the cards to compose devices.
5. Click **OK**.

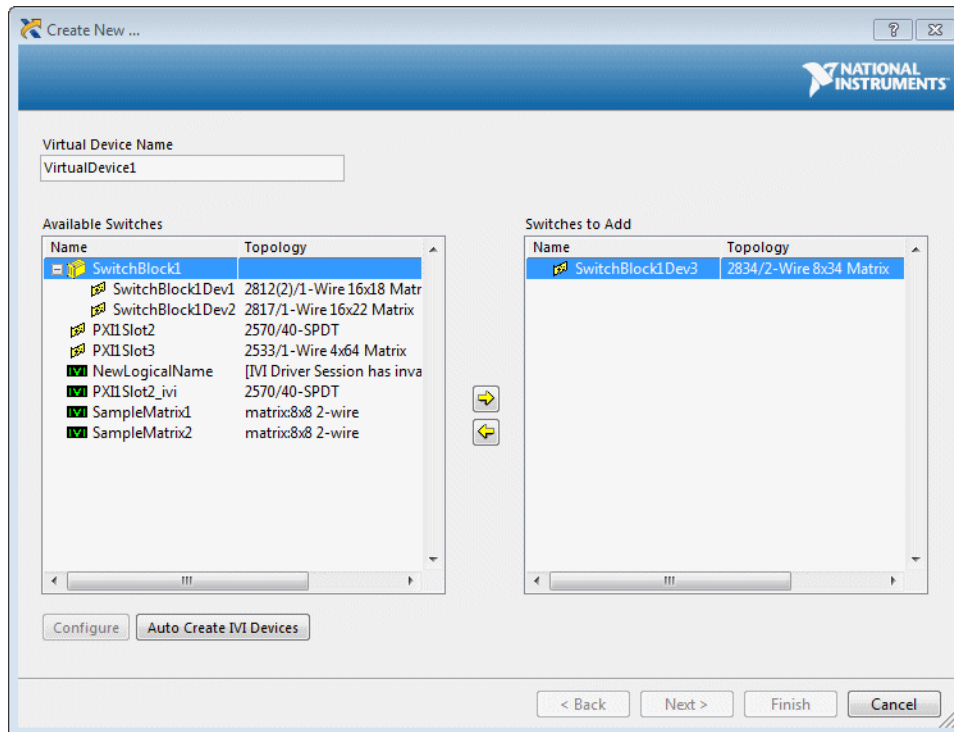


2. Create a New Virtual Device


Complete the following steps to create a new virtual device:

1. Launch NI MAX with NI Switch Executive.

2. Expand **Devices and Interfaces**.
3. Right-click **NI Switch Executive Virtual Devices** and select **Create New**. The Create New dialog box opens.
4. Type a name in the **Virtual Device Name** textbox or accept the predefined Virtual Device Name. The name you select is the name used to refer to the virtual device in the API.

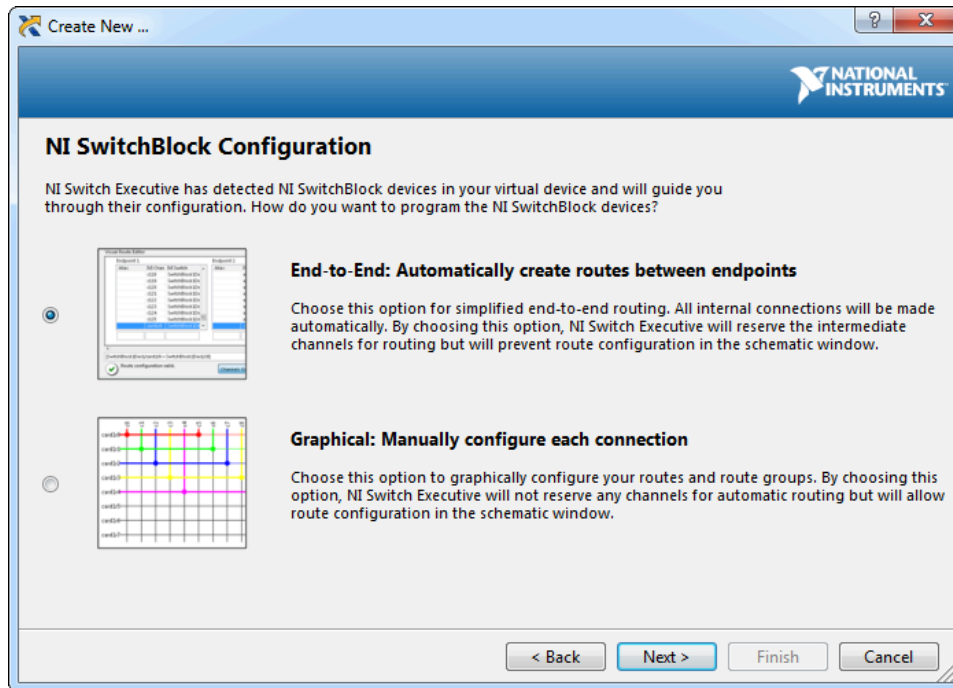


Note Some characters are reserved and cannot be used in virtual device names.

5. Select a switch device or multiple switch devices from the **Available Switches** listbox and click  to add each switch to the **Switches to Add** listbox.
6. (Optional) Click **Auto Create IVI Devices** to create default IVI configurations for all NI Switches detected in your system. IVI configurations are required only if you want to import a configuration from NI Switch Executive 3.0 or earlier. In NI Switch Executive 3.5 and later, NI switch devices do not require the use of IVI.
7. Click **Next**. A dialog box displays the status of the virtual device creation.
8. Click **Next**. The NI SwitchBlock Configuration dialog box launches to take you through configuration of your new NI SwitchBlock virtual device.
9. Select one of the following programming preferences and click **Next**:
 - **End-to-End**—Configuration automatically creates routes between endpoints.

All internal connections are made automatically. NI Switch Executive reserves the intermediate channels for routing but prevents route configuration in the Schematic window.

- **Graphical**—Configuration allows you to manually configure your routes and route groups in the Schematic window of the Schematic Configuration tab. NI Switch Executive does not reserve any channels for routing.



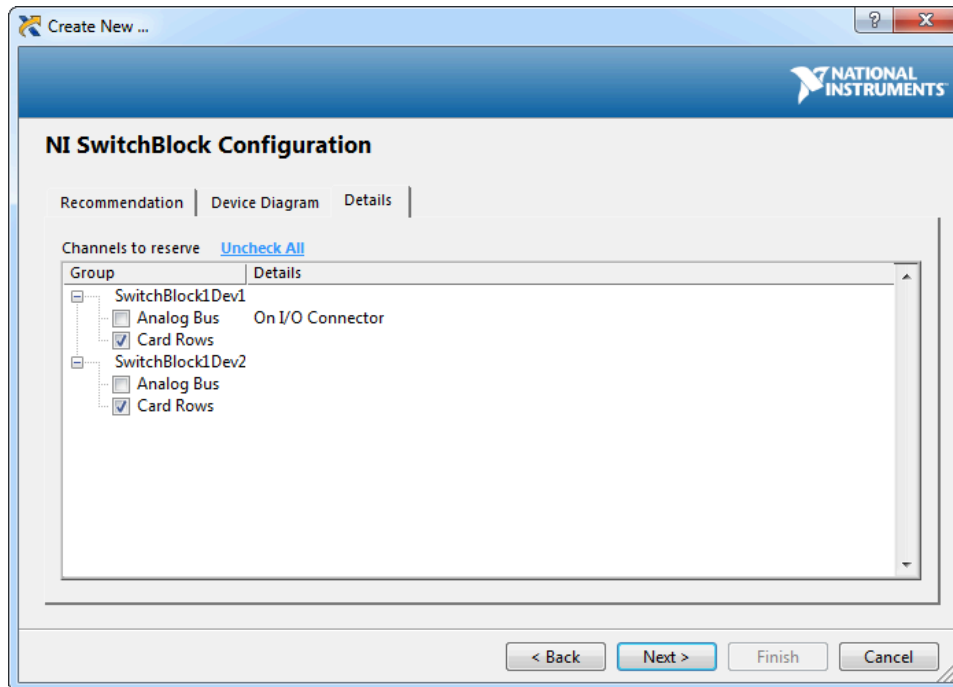
10. (End-to-End configuration only) Reserve channels for routing.

If you chose End-to-End configuration, a channel reservation window displays the following three tabs:

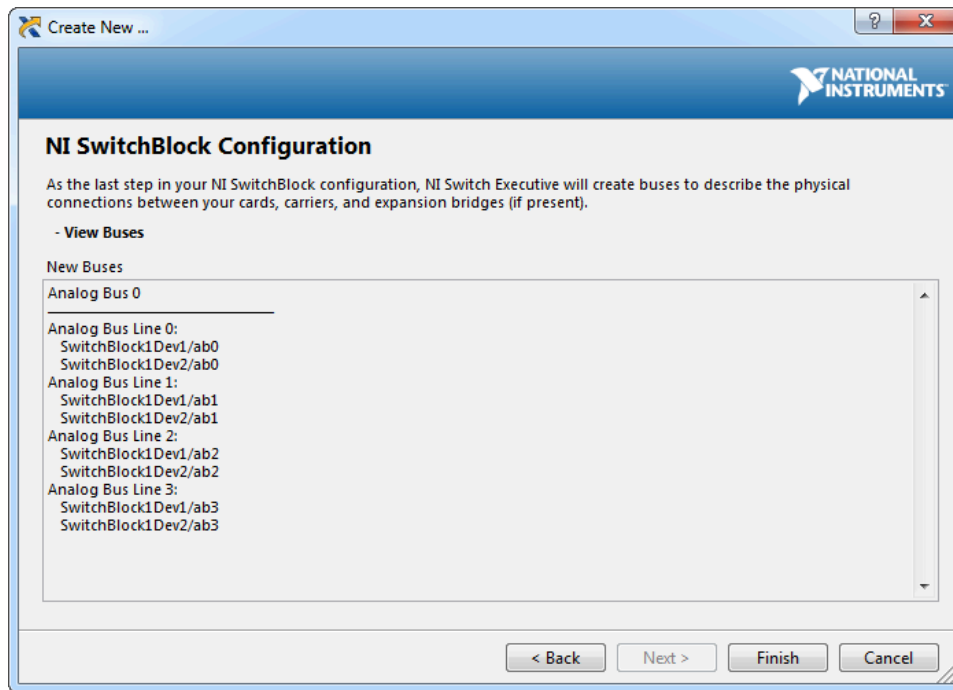
- **Recommendation**—Displays the recommended configuration
- **Device Diagram**—Displays a device diagram showing the channels to help you make connections
- **Details**—Provides options for channel reservation with the following items:
 - **Channels to Reserve**—Select specific buses and rows for channel reservation.
 - **Uncheck All**—Deselects all buses and rows.
 - **Use Defaults**—Reserves default rows and buses. This item is only visible when **Uncheck All** has been clicked.

By default, On I/O Connectors are not reserved. If you reserve an On I/O Connector, you cannot use that channel as a route endpoint.

1. Select buses and rows for channel reservation.
2. Click **Next**.




11. Create buses.
 1. Click **View Buses** to see the buses that NI Switch Executive will create to describe the physical connections between your cards, carriers, and expansion bridges (if present).
 2. Click **Finish** to create the buses. MAX automatically browses to the NI Switch Executive virtual device so you can continue with system configuration.



Creating a Virtual Device Using the Switch SFP

Complete the following steps to create an NI Switch Executive virtual device using the Switch Soft Front Panel (SFP):

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces**.
3. Right-click the DAQmx switch from which you want to create a virtual device and select **Test Panels**. The Switch SFP opens.
4. Select a device from the **Device** drop-down listbox.
5. Select a topology for the device from the **Topology** drop-down listbox.
6. Click , **Create NI Switch Executive Virtual Device**. MAX automatically browses to the NI Switch Executive virtual device with a predefined name of `VirtualDeviceX` under **Devices and Interfaces»NI Switch Executive Virtual Devices**.



Note At this time, NI Switch Executive virtual device creation in the Switch SFP is not supported for NI SwitchBlock devices.

Configuring a Virtual Device

Use [schematic](#) and/or [extended](#) configuration in MAX to configure an NI Switch

Executive virtual device.



The following table lists the configuration tasks you can perform using schematic configuration and extended configuration:

	Schematic Configuration	Extended Configuration
Add Channel Aliases	√	√
Add/Remove Route Groups	√	√
Nest Route Groups	√	√
Add/Remove Routes	√	√
Add/Remove Hardwires	√	√
Add/Remove Exclusions	N/A	√
Add/Remove Buses	N/A	√

Setting Configuration Options

When selecting channel paths, the list of available channels can become cumbersome in large systems. To reduce clutter, a number of configuration options are available.

Complete the following steps to configure options in NI Switch Executive:


1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Right-click the virtual device you want to configure and select **Options**, or click  on the NI Switch Executive taskbar and select  **Options**. The Configuration Options dialog box opens.
4. Click the **General** tab.
 1. Configure the channel name display. The following table lists the available options:

Option	Description
Display full name only	NI Switch Executive lists the full name of the channel. For example, SampleMatrix1/ab0.

Display alias or full name (default)	If the channel has an alias, the alias is listed. If no alias is designated for the channel, the full name is listed.
Display alias and full name	NI Switch Executive lists the alias and the full name of the channel.

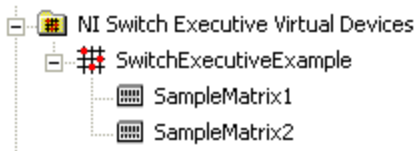
2. Configure the visual route editor selections. The following table lists the available options:

Option	Description
Hide channels without aliases	NI Switch Executive only lists channels with aliases. NI Switch Executive disables this option by default.
Defer updates on configuration changes	NI Switch Executive defers verification and rerouting when configuration changes are made (changes to names, reserved for routing channels, and so on). In systems with a large number of routes, this verification and rerouting process can be time consuming. To speed up this operation, you can defer this process and validate your configuration at a later time. To reroute any broken routes, select the broken routes and click Retry . NI Switch Executive disables this option by default.
Routing search depth	NI Switch Executive chooses the depth by which the routing algorithm searches for paths to recommend changes to reserved for routing channels. You can select Low (default), Medium , or High .

5. Click the **Schematic** tab and select **Disable delete notification dialog** to turn off delete notifications in the Schematic Configuration tab.
6. Click **OK**.
7. Click  **Save** on the NI Switch Executive taskbar to save the changes.


Configuring IVI Switches

IVI switches display in the configuration tree beneath the [NI Switch Executive virtual device](#).



Adding IVI Switches


Complete the following steps to add an IVI switch to an existing NI Switch Executive virtual device:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Right-click the virtual device that you want to modify and select **Add IVI Switch**. The Add Device dialog box appears.
4. Select the switches you want to add from the **Available Switches** listbox and click  to move them to the **Switches to Add** listbox.



Note If you are using National Instrument switches, click **Configure** to configure the settings of those switches before you add them to the **Switches to Add** listbox.

If you are using [third-party switches](#), select the IVI logical name you created.

5. Click **Next**. The new device is added to your configuration.
6. Click **Finish**.
7. Click  on the NI Switch Executive taskbar to save the changes.

Refer to [Configuring Channels](#) to continue configuring your NI Switch Executive virtual device.

Removing IVI Switches

Complete the following steps to remove an IVI switch from an existing NI Switch Executive virtual device:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Expand the NI Switch Executive virtual device containing the IVI switch you want to

remove.

4. Right-click the IVI switch to remove and select **Remove IVI Switch**. A dialog box opens to confirm the removal of the IVI switch.



Note When you remove an IVI switch, its channels are also removed from your configuration. This removal may break some of your routes.

5. Click **Yes** to remove the IVI switch or **No** to abort.

Using Third-Party Switches

NI Switch Executive uses IVI driver sessions and/or IVI logical names as a way to communicate with all IVI-compliant switches. For NI Switch Executive to recognize a third-party switch, you must create an IVI configuration.


Complete the following steps to create an IVI configuration for a third-party, IVI-compliant switch:

Create a Driver Session


1. Launch NI MAX with NI Switch Executive.
2. Expand **IVI Drivers**.
3. Right-click **Driver Sessions**.
4. Select **Create New (case-sensitive)**. A driver session with a default name of `NewDriverSessionX` is created.
5. Select the **General** tab at the bottom of the window that displays on the right.
6. Edit the driver setup string as needed. You may need to modify the driver setup string to specify appropriate topology information. Refer to your third-party software vendor for more information.
7. Select the **Hardware** tab.
8. Select the hardware in the **Hardware Assets** window.
9. (Optional) Click the **Add** and **Remove** buttons to add and remove hardware assets.
10. Edit the resource descriptor as needed. Refer to the instrument driver vendor for information about specifying the resource descriptor.
11. Select the **Software** tab.
12. Select the IVI driver for the switch from the **Software Module** drop-down listbox.



Note If the software module does not display in the drop-down listbox, contact the instrument driver vendor for an updated installer.

13. Cycle through the tabs and make any additional changes where appropriate.
14. Click  Save IVI Configuration in the NI Switch Executive taskbar to save the changes.

Create a Logical Name

1. Launch NI MAX with NI Switch Executive.
2. Expand **IVI Drivers**.
3. Right-click **Logical Names**.
4. Select **Create New (case-sensitive)**. A logical name with a default name of `NewLogicalName` is created.
5. Select the driver session you created for your switch from the **Driver Session** drop-down listbox in the window on the right.
6. Click  Save IVI Configuration in the NI Switch Executive taskbar to save the changes.

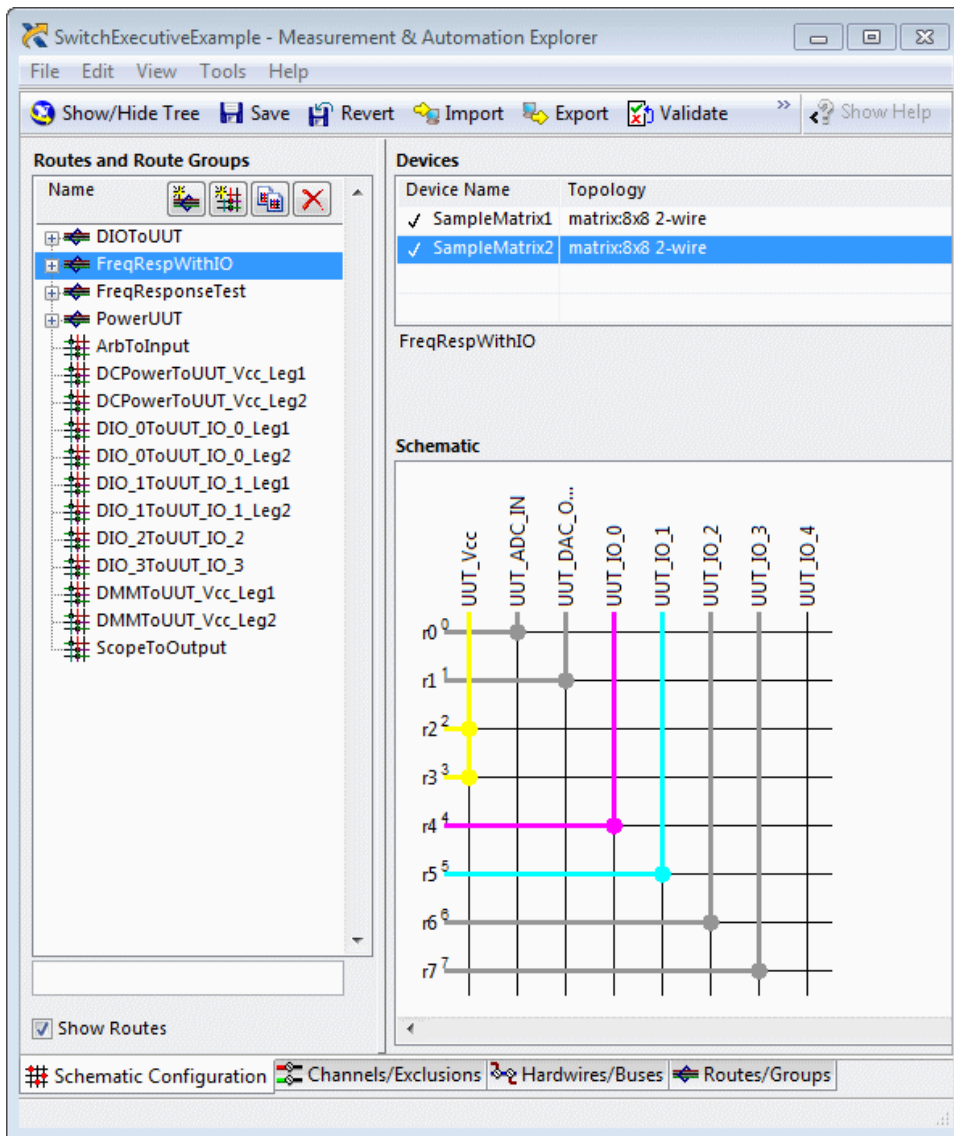
Schematic Configuration

In the **Schematic Configuration** tab, NI Switch Executive renders a schematic, or a graphical representation, of each switch in a virtual device. Click the device schematic to quickly configure a virtual device, including channel aliases and hardwires.



Note NI Switch Executive represents [third-party](#), IVI-compliant switch modules and [independent topologies](#) in the **Schematic Configuration** tab as channel listboxes.

The **Routes and Route Groups** and **Devices** listboxes and the **Schematic** window compose the **Schematic Configuration** tab. In the following figure, a matrix topology schematic displays in the Schematic window:







Routes and Route Groups

The **Routes and Route Groups** listbox contains all the route groups associated with a virtual device.



Note If the virtual device does not have any existing route groups defined, NI Switch Executive creates a default route group, `RouteGroup0`, in the **Routes and Route Groups** listbox to which you can add a route.

Use the buttons at the top of the **Routes and Route Groups** listbox to manage route groups and routes.

Button	Function
	Adds a route group to the virtual device and the Routes and Route Groups listbox. When routes are added to the route group, the new route group displays in the Schematic window.
	Adds a route to the virtual device and the Routes and Route Groups listbox. This button is disabled when the Show Routes checkbox is disabled.
	Creates a copy of the route group and adds it to the virtual device and the Routes and Route Groups listbox. The copy of the route group displays in the Schematic window. When the Show Routes checkbox is enabled, this button can copy routes in addition to route groups.
	Deletes the route group from the virtual device and the Routes and Route Groups listbox. The route group no longer displays in the Schematic window. When deleting a nested route group, this button disassociates the child route group from the parent route group. When the Show Routes checkbox is enabled, this button can delete routes in addition to route groups.



Tip You can also right-click any name in the **Routes and Route Groups** listbox to manually manage the routes/route groups.

Show Routes

Enable the **Show Routes** checkbox, located below the **Routes and Route Groups** listbox, to manage the individual routes of a route group. When you enable the **Show Routes** checkbox, the create route button is enabled, and you can use the copy button to copy a route.

Filter

Use the filter textbox, located directly below the **Routes and Route Groups** listbox, for searching. Type the first characters of a route or route group name, and all names that match the search string display in the **Routes and Route Groups** listbox. The search string is not case sensitive.

Devices

The **Devices** listbox contains all the IVI switches associated with a virtual device. Click a switch device to display the channels, routes, route groups, and hardwires associated

with the switch in the **Schematic** window.

A checkmark displays to the left of the device name in the **Devices** listbox to indicate that a route exists on the device.

Schematic

The **Schematic** window is a graphical representation of all the channels, routes, and hardwires of a route group on a particular NI switch device associated with a virtual device.

When you associate a channel with a hardwire, the hardwire number appends as a superscript to the channel name in the schematic.


A route displays in color on the schematic to represent its electrical connectivity. If an electrical path is shared between two or more routes, the routes display in the same color on the schematic.



Note A route in the schematic that appears dimmed denotes that the route is not configurable in the **Schematic Configuration** tab.

Getting Started with Schematic Configuration

The following steps represent the general workflow for schematic configuration:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Click  at the top of the **Routes and Route Groups** listbox to [create a route group](#).
5. Select the device that contains the channel to configure in the **Devices** listbox. The device schematic displays in the **Schematic** window.
6. Click a channel name in the **Schematic** window to access the **Edit Alias & Hardwire** dialog box and [configure channel aliases and hardwires](#).

7. Repeat step 6 until you have configured all the channel aliases and hardwires on the virtual device.
8. Complete one of the following steps depending on the topology of the switch device selected to [create a route](#) between channels:
 - **General purpose topology**—Click a crosspoint between two channels on the specific relay you want to use in the **Schematic** window.
 - **Matrix topology**—Click a crosspoint between two channels in the **Schematic** window.
 - **Multiplexer topology**—Click a channel connection point in the **Schematic** window.
 - **Third-party IVI-switch and [independent](#) topologies**—Select one channel from the **Channel 1** listbox and one channel from the **Channel 2** listbox and click **Create**.

The route displays in the **Schematic** window and is added to the route group.




Note You can click the point of intersection between two channels, crosspoint or channel connection point, in the **Schematic** window to [delete the route](#) between channels.

9. Repeat step 8 until you have created all the routes that you want to add to the route group. You can select a different device in the **Devices** listbox to create routes across multiple devices.



Note By default, NI Switch Executive does not display the routes associated with a route group in the **Routes and Route Groups** listbox. Enable the **Show Routes** checkbox below the **Routes and Route Groups** listbox to display the routes associated with a route group and manage those routes individually.


10. Click  **Save** on the NI Switch Executive taskbar to save the changes.
11. [Validate](#) the changes.

Adding a Route Group

Complete the following steps to add a route group:


1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the

Schematic Configuration tab.

- Click  at the top of the **Routes and Route Groups** listbox. NI Switch Executive creates an empty route group with a default name, `RouteGroupX`.




Note If the virtual device does not have any route groups defined, NI Switch Executive creates a default route group, `RouteGroup0`, in the **Routes and Route Groups** listbox.

- (Optional) Right-click the route group and select **Rename**, or click the route group and press <F2> to edit a route group name.
- Click  on the NI Switch Executive taskbar to save the changes.

Copying a Route Group

Complete the following steps to copy a route group:

- Launch NI MAX with NI Switch Executive.
- Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
- Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
- Click the route group you want to copy in the **Routes and Route Groups** listbox.
- Click . NI Switch Executive creates a copy of the route group, with a predefined name, in the **Routes and Route Groups** listbox and displays the copied route group in the **Schematic** window.




Note NI Switch Executive shares associated routes between a route group and its copy.

- (Optional) Add routes to fill or disassociate routes to empty the route group.
- Click  on the NI Switch Executive taskbar to save the changes.

Nesting a Route Group

Complete the following steps to nest `RouteGroup0` in `RouteGroup1`, where `RouteGroup0` and `RouteGroup1` are variables:

- Launch NI MAX with NI Switch Executive.


2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Select `RouteGroup0` in the **Routes and Route Groups** listbox.
5. Drag and drop `RouteGroup0` on `RouteGroup1`. NI Switch Executive nests `RouteGroup0`—the child route group—within `RouteGroup1`—the parent route group—and displays the routes of both in the schematic of `RouteGroup1`. Any edits that you make in the child route group are reflected in the parent route group.
6. Click  on the NI Switch Executive taskbar to save the changes.

Disassociating a Nested Route Group



Note Disassociating a nested route group from a parent route group only removes the route group from the parent route group and does **not** delete the route group from the **Routes and Route Groups** listbox. If you want to delete a nested route group from its parent route group **and** remove it from the **Routes and Route Groups** listbox, delete the nested route group instead.

Complete the following steps to disassociate a nested route group:


1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Expand the route group that contains the nested route group you want to disassociate in the **Routes and Route Groups** listbox.
5. Right-click the route group you want to disassociate and select **Disassociate**. A dialog box opens to confirm the disassociation.
6. Click **OK**. NI Switch Executive disassociates the nested, child route group from its parent route group. Notice that the previously nested route group is still available in the **Routes and Route Groups** listbox.
7. Click  on the NI Switch Executive taskbar to save the changes.

Deleting a Route Group




Note Deleting a route group removes it from the virtual device: that is, NI Switch Executive deletes the route group **and** removes it from the **Routes and Route Groups** listbox.

Complete the following steps to delete a route group:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Select the route group you want to delete in the **Routes and Route Groups** listbox and click , or right-click the route group you want to delete and select **Delete**. A dialog box opens to confirm the deletion.




Note When you delete a parent route group, NI Switch Executive disassociates all nested route groups. Although NI Switch Executive deletes the parent route group from the virtual device and the **Routes and Route Groups** listbox, the nested, child route groups remain available in the **Routes and Route Groups** listbox.

5. Click **OK** to delete the route group. NI Switch Executive deletes the route group from the virtual device and the **Routes and Route Groups** listbox.
6. Click  **Save** on the NI Switch Executive taskbar to save the changes.

Adding a Route

Complete the following steps to add a route:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Click the switch device to which you want to add a route in the **Devices** listbox. NI Switch Executive displays the device schematic in the **Schematic** window.
5. Enable the **Show Routes** checkbox below the **Routes and Route Groups** listbox.
6. Complete one of the following steps to add a route:

- Add a route **not associated** with a route group—Click  above the **Routes and Route Groups** listbox.



Tip You can also right-click in the **Routes and Route Groups** listbox and select **Add Route»Empty Route**.

- Add a route **associated** with a route group—Click the route group to which you want to add a route in the **Routes and Route Groups** listbox. You can add a route to an existing route group or [create a new route group](#) to which to add a route.




Note If the virtual device does not have any existing route groups defined, NI Switch Executive creates a default route group, `RouteGroup0`, in the **Routes and Route Groups** listbox to which you can add a route.

7. Complete one of the following steps depending on the topology of the switch device selected to create a route between channels:
 - **General purpose topology**—Click a crosspoint between two channels on the specific relay you want to use in the **Schematic** window.
 - **Matrix topology**—Click a crosspoint between two channels in the **Schematic** window.
 - **Multiplexer topology**—Click a channel connection point in the **Schematic** window.
 - **Third-party IVI-switch and [independent](#) topologies**—Select one channel from the **Channel 1** listbox and one channel from the **Channel 2** listbox and click **Create**.

The route displays in the **Schematic** window and is added to the **Routes and Route Groups** listbox. A checkmark displays to the right of the device name in the **Devices** listbox to indicate a route exists on the device.



Note Clicking on the same crosspoint or channel connection point in a different route group creates a new route on the virtual device.

8. (Optional) [Edit](#) the route name.
9. Click  on the NI Switch Executive taskbar to save the changes.

Editing a Route


Complete the following steps to edit a route:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Enable the **Show Routes** checkbox under the **Routes and Route Groups** listbox to view all routes.
5. Click the route you want to edit in the **Routes and Route Groups** listbox.



Note When you edit a [shared route](#), the edits apply to every instance of the route.

6. Complete one or more of the following editing tasks:


Copy	Click  to create a copy of the route. NI Switch Executive creates a copy of the route, with a predefined name, in the Routes and Route Groups listbox.
Rename	Press <F2> or right-click the route and choose Rename to rename the route.
Edit the Path	<ul style="list-style-type: none"> ◦ General purpose, matrix, and multiplexer topologies—Click <i>another</i> point of intersection between two channels in the Schematic window. ◦ Third-party IVI-switch and independent topologies—Select one channel from the Channel 1 listbox and one channel from the Channel 2 listbox and click Change. <p>NI Switch Executive deletes the original route path and creates a new route path. If you want to create the route on a different device, click the device in the Devices listbox before you edit the path.</p>

7. Click  on the NI Switch Executive taskbar to save the changes.

Sharing a Route Between Route Groups

Complete the following steps to share a route, `routeA`, between two route groups, `Ro`

uteGroup0 and RouteGroup1—where routeA, RouteGroup0, and RouteGroup1 are variables:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Enable the **Show Routes** checkbox under the **Routes and Route Groups** listbox. NI Switch Executive displays all the routes of the virtual device, associated in route groups or not, in the **Routes and Route Groups** listbox.
5. Click routeA in the **Routes and Route Groups** listbox.
6. Drag routeA and drop it on RouteGroup0.
7. Click routeA in the **Routes and Route Groups** listbox.
8. Drag routeA and drop it on RouteGroup1. routeA is now shared between RouteGroup0 and RouteGroup1.
9. Click  on the NI Switch Executive taskbar to save the changes.

Disassociating a Route from a Route Group



Note Disassociating a route from a route group only removes the route from its associated route group. The route remains on the virtual device and in the **Routes and Route Groups** listbox. If you want to delete a route from all its associated route groups **and** the **Routes and Route Groups** listbox, delete the route instead.

Complete the following steps to disassociate a route from a route group:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Enable the **Show Routes** checkbox under the **Routes and Route Groups** listbox to view all routes.
5. Expand the route group that contains the route you want to disassociate in the **Routes and Route Groups** listbox.
6. Right-click the route you want to disassociate and select **Disassociate**. A dialog box opens to confirm the disassociation.
7. Click **OK**. NI Switch Executive disassociates the route from its associated route

group. Notice that the route is still available in the **Routes and Route Groups** listbox.

- Click  on the NI Switch Executive taskbar to save the changes.

Deleting a Route



Note Deleting a route from a virtual device removes the route from its associated route groups **and** the **Routes and Route Groups** listbox. If you want to remove a route from a route group but keep it available in the **Routes and Route Groups** listbox, [disassociate](#) the route instead.




Note To delete a route from its grandparent route group, you must delete the route from its parent route group.

Complete the following steps to delete a route:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Click the device that contains the route that you want to delete in the **Devices** listbox. NI Switch Executive displays the device schematic in the **Schematic** window.
5. Enable the **Show Routes** checkbox below the Routes and Route Groups listbox.
6. Click the route to delete in the **Routes and Route Groups** listbox.




Note If the route you want to delete is shared among multiple route groups, selecting the route within the route group will prompt you to disassociate the route, rather than delete it. To delete the route, select it as shown in the listbox not nested under any route groups.

7. Click  or right-click the route and select **Delete** to delete the route.




Tip With the parent route group selected in the **Routes and Route Groups** listbox, you can delete a route associated with a single route group by clicking the point of intersection—crosspoint or channel connection point—between two channels in the **Schematic** window.

For third-party IVI-switch and independent topologies, with the parent route group selected in the **Routes and Route Groups** listbox, you can delete a route associated with a single route group by selecting the route from the **Connections** listbox and clicking **Delete**.

8. Click **OK** to delete the route.
9. Click  **Save** on the NI Switch Executive taskbar to save the changes.



Configuring a Channel

Complete the following steps to configure a channel:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Click the device that contains the channel that you want to configure in the **Devices** listbox. NI Switch Executive displays the device schematic in the **Schematic** window.
5. Click a channel in the **Schematic** window to access the **Edit Alias & Hardware** dialog box.
6. Type a new alias name or edit the existing alias of the channel in the **Alias Name** textbox.
7. Select the **Hardwires** tab to designate whether the channel is associated with a hardwire.
 1. Select the hardwire that you want to associate with the channel, or create a new hardwire to associate, in the **Available Hardwires** listbox.
 2. Click **Associate *channelName* with *hardwireName***, where ***channelName*** and ***hardwireName*** are variables, depending on the channel and hardwire you want to associate. NI Switch Executive adds the channel to the Associated Channels listbox.
8. Click **OK** to accept the configuration. The hardwire number appends as a superscript to the channel name in the **Schematic** window.
9. Click  **Save** on the NI Switch Executive taskbar to save the changes.

Adding a Hardware

Complete the following steps to add a hardware:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Verify that the required switch device is selected in the **Devices** listbox and its schematic appears in the **Schematic** window.
5. Click a channel in the **Schematic** window to access the **Edit Alias & Hardware** dialog box.
6. Type an alias name for the channel in the **Alias Name** textbox.
7. Select the **Hardwires** tab.
8. Click  in the **Available Hardwires** listbox. NI Switch Executive adds a new hardware, with a predefined name, to the Available Hardwires listbox. To edit a hardware name, click the hardware name and press <F2>.
9. (Optional) Click **Associate *channelName* with *hardwareName***, where ***channelName*** and ***hardwareName*** are variables, depending on the channel and hardware you want to associate. NI Switch Executive adds the channel to the Associated Channels listbox.
10. Click **OK**. The hardware number appends as a superscript to the channel name in the **Schematic** window.
11. Click  on the NI Switch Executive taskbar to save the changes.

Deleting a Hardware





Note Deleting hardwires can break routes requiring the hardware. Always validate a virtual device after you delete a hardware.

Complete the following steps to delete a hardware:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the

Schematic Configuration tab.

4. Verify that the required switch device is selected in the **Devices** listbox and its schematic appears in the **Schematic** window.
5. Click a channel in the **Schematic** window to access the **Edit Alias & Hardwire** dialog box.
6. Select the **Hardwires** tab.
7. Select the hardwire you want to delete in the **Available Hardwires** listbox and click . A dialog box opens to confirm the deletion.
8. Click **Yes** to delete the hardwire. NI Switch Executive deletes the hardwire from the Available Hardwires listbox and disassociates any channels associated with the hardwire.
9. Click **OK**.
10. Click  on the NI Switch Executive taskbar to save the changes.

Extended Configuration Tasks

Some NI Switch Executive configuration tasks are only accessible in the [extended configuration](#) tabs in MAX. These tasks include:

- Reserving a channel for routing



Note When you create a route with reserved for routing channels in the extended configuration tabs, NI Switch Executive dims the route in the Schematic window. A dimmed route denotes that the route is not configurable in the Schematic Configuration tab.

- Disabling a channel
- Configuring exclusions
- Configuring buses

Although these tasks are not accessible in the Schematic Configuration tab, they are preserved in the virtual device configuration.

Third-Party Switches

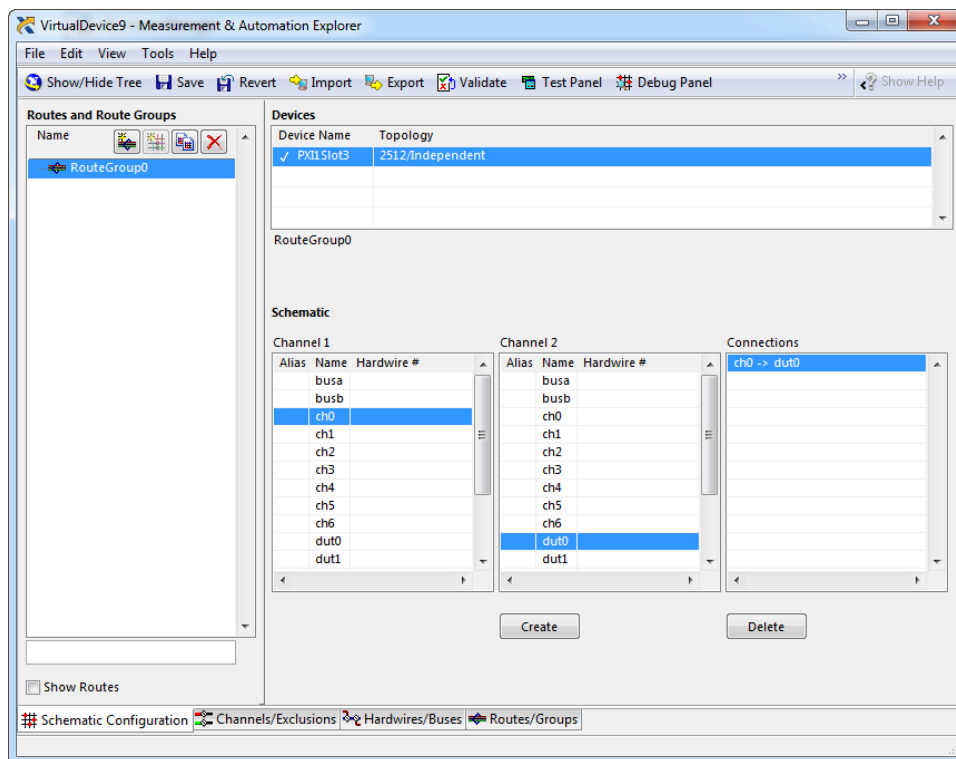
In addition to NI switch modules, NI Switch Executive virtual devices support IVI-

compliant, [third-party](#) switches. Because third-party switches do not contain the information required to draw a schematic, NI Switch Executive displays the schematic as three listboxes: Channel 1, Channel 2, and Connections.

NI Switch Executive also displays schematic listboxes for NI switches that use an independent topology. Refer to [Independent Topologies](#) for instructions about how to configure a third-party switch using the **Schematic Configuration** tab.

Independent Topologies

Some NI switch modules can be configured with an independent topology. Like [third-party](#) switch modules, NI switch modules using an independent topology do not contain the information required to draw a schematic. Instead, NI Switch Executive displays the schematic as three listboxes, **Channel 1**, **Channel 2**, and **Connections**, as shown in the following figure:



The **Routes and Route Groups** and **Devices** listboxes function the same in an independent topology as they do in other topologies. In an independent topology, the **Channel 1** and **Channel 2** listboxes display all the channels available for the device, including the aliases and hardwires associated with the channels. The **Connections**

listbox displays the routes associated with a route group.

Use the following instructions to configure an NI switch module with an independent topology using the **Schematic Configuration** tab.



Note You can also use the following instructions to configure an IVI-compliant, third-party switch module using the **Schematic Configuration** tab.

Route Groups

Schematic configuration of a route group in an independent topology is similar to schematic configuration of a route group in other topologies. Refer to the following related topics for information about configuring a route group in an independent topology:

- [Adding a Route Group](#)
- [Copying a Route Group](#)
- [Nesting a Route Group](#)
- [Disassociating a Nested Route Group](#)
- [Deleting a Route Group](#)

Routes


Schematic configuration of a route in an independent topology has similarities and differences with schematic configuration of a route in other topologies. Refer to the following related topics for information about configuring a route in an independent topology:

- Adding a Route

Complete the following steps to add a route to an independent topology:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces**»**NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify and navigate to the **Schematic Configuration** tab.
4. Select the device with the independent topology to which you want to add a

route in the **Devices** listbox.

5. Enable the **Show Routes** checkbox below the Routes and Route Groups listbox.
6. Complete one of the following steps to add a route:
 - **Add a route *not associated* with a route group**—Click  above the **Routes and Route Groups** listbox.



Tip You can also right-click in the **Routes and Route Groups** listbox and select **Add Route»Empty Route**.

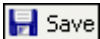
- **Add a route *associated* with a route group**—Click the route group to which you want to add a route in the **Routes and Route Groups** listbox. You can add a route to an existing route group or [create a new route group](#) to which to add a route.



Note If the virtual device does not have any existing route groups defined, NI Switch Executive creates a default route group, `RouteGroup0`, in the **Routes and Route Groups** listbox to which you can add a route.

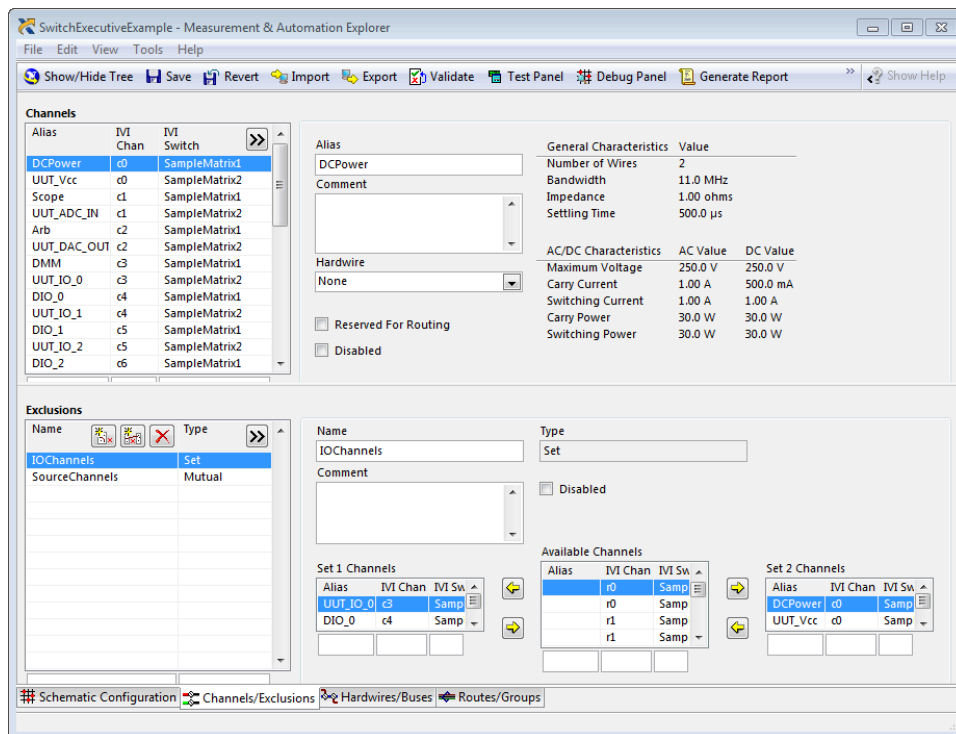
7. Select a channel in the Channel 1 listbox as an endpoint for the route.
8. Select a channel in the Channel 2 listbox as another endpoint for the route.
9. Confirm the route creation:
 - If the route selected is ***not associated*** with a route group, click **Change**.
 - If the route selected is ***associated*** with a route group, click **Create**.

NI Switch Executive adds a route with a predefined name to the **Routes and Route Groups** listbox, adds a checkmark to the left of the device name in the **Devices** listbox to indicate a route exists on the device, and displays the path of the route in the **Connections** listbox.

10. Click  on the NI Switch Executive taskbar to save the changes.
 - [Editing a Route](#)
 - [Sharing a Route](#)
 - [Disassociating a Route](#)
 - [Deleting a Route](#)

Extended Configuration

Use the Extended Configuration tabs—**Channels/Exclusions**, **Hardwires/Buses**, and **Routes/Groups**—to navigate through the configuration of an NI Switch Executive virtual device and configure channels, exclusions, hardwires, buses, route groups, and routes.







Batch Renaming

Working with systems of large channel counts may require you to rename large selections of channels, routes, route groups, and so on. NI Switch Executive offers a method for renaming large NI Switch Executive virtual device objects with a sequential naming scheme. For example, you can select a large number of objects and rename them `Object_X`.


Complete the following steps to rename a set of objects:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify.

4. Click the [extended configuration](#) tab that contains the objects you want to rename.
5. Select the objects you want to rename. You can hold down <Shift> or <Ctrl> and click to select multiple objects.
6. Click  Batch Rename on the NI Switch Executive taskbar. If this button is not visible, click  to expand the taskbar. The Batch Rename dialog box opens.
7. Use  and  to order the objects in the **Items to Rename** listbox. NI Switch Executive renames objects sequentially based on their order in this list.



Note The initial order of the objects in the **Items to Rename** listbox reflects the order in which the objects were selected.

8. Type a base name for the objects in the **Base Name** textbox. NI Switch Executive uses this name as the prefix for all the selected objects.
9. Type a number in the **Start Index** textbox. NI Switch Executive appends this number to the name of the first object in the **Items to Rename** listbox. For each object thereafter, NI Switch Executive increments the number by one and appends it to the name.
10. Click **Okay**. NI Switch Executive batch renames the objects based on the values you entered in the **Base Name** and **Start Index** textboxes. For example, with a base name of `Object` and a start index of `1`, NI Switch Executive renames the objects `Object1`, `Object2`, `Object3`, and so on.
11. Click  Save on the NI Switch Executive taskbar to save the changes.

Sorting, Filtering, & Commenting

You can sort, filter, and add comments to a particular element of an NI Switch Executive virtual device in the [extended configuration](#) tabs.

Sorting

Click the heading of a column in the listbox of any object to sort the list alphanumerically.

Filtering

Use the textboxes below every column under an object's listbox to filter the contents of

that column. For example, if you type **C** in the filtering textbox, NI Switch Executive filters out every item in the list that does not begin with a **C**. Entering **C1** filters out every item that does not begin with **C1**, and so on. You can use multiple filters in a textbox by typing commas. For example, entering **R, C** filters all items that begin with **R** or **C**. Filtering is **not** case sensitive.

Comments

Use the Comment textboxes to document the virtual device. Choose the item to comment, and type your comments.

Channels/Exclusions Tab



Caution Connecting one signal input to another signal input can create a potentially hazardous and destructive situation. Refer to [Exclusion](#) for more information.

Use the **Channels/Exclusions** tab to configure channels and define exclusions in NI Switch Executive.

The screenshot shows the NI Switch Executive software interface. The main window is titled "SwitchExecutiveExample - Measurement & Automation Explorer". The interface is divided into several sections:

- Channels:** A list of channels with columns for Alias, IVI Chan, and IVI Switch. The list includes items like DCPower, UUT_Vcc, Scope, UUT_ADC_IN, Arb, UUT_DAC_OUT, DMM, UUT_IO_0, DIO_0, UUT_IO_1, DIO_1, UUT_IO_2, and DIO_2.
- General Characteristics:** A table showing properties for the selected channel (DCPower):

General Characteristics	Value
Number of Wires	2
Bandwidth	11.0 MHz
Impedance	1.00 ohms
Settling Time	500.0 μ s
- AC/DC Characteristics:** A table showing AC and DC characteristics for the selected channel:


AC/DC Characteristics	AC Value	DC Value
Maximum Voltage	250.0 V	250.0 V
Carry Current	1.00 A	500.0 mA
Switching Current	1.00 A	1.00 A
Carry Power	30.0 W	30.0 W
Switching Power	30.0 W	30.0 W
- Exclusions:** A section for defining exclusions. It includes a list of exclusions (IOChannels, SourceChannels) and a detailed configuration area for a selected exclusion (IOChannels). The configuration area includes fields for Name, Type, Comment, and a checkbox for Disabled. It also features "Available Channels" and "Set 1 Channels" / "Set 2 Channels" sections for selecting specific channels.

Configuring Channels

When adding a new device to the virtual device, NI Switch Executive automatically queries the switch and adds its channels, and their specifications, to the channel list. Each channel has a list of predefined attributes, as shown in the following figure:

General Characteristics	Value	
Number of Wires	2	
Bandwidth	11.0 MHz	
Impedance	1.00 ohms	
Settling Time	500.0 μ s	
AC/DC Characteristics	AC Value	DC Value
Maximum Voltage	250.0 V	250.0 V
Carry Current	1.00 A	500.0 mA
Switching Current	1.00 A	1.00 A
Carry Power	30.0 W	30.0 W
Switching Power	30.0 W	30.0 W



Tip If the attribute table is not visible, click .

Complete the following steps to configure channels:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify.
4. Click the **Channels/Exclusions** tab.
5. Select a channel to configure.
6. Accept the default alias of the channel, or modify it in the **Alias** textbox.



Note Some characters are reserved and cannot be used in channel names.


7. Type any comments you may have in the **Comment** textbox.



Tip Use the **Comment** textbox for organizational purposes. Place keywords or letters at the beginning of a comment for convenient sorting and filtering. Document information such as wire gauge and color.

- Designate whether the channel is connected to a hardware by selecting from the **Hardware** drop-down listbox, as shown in the following figure:



If the **Hardware** drop-down listbox is not visible, click .

Hardwires can also be [defined](#) in the [hardware panel](#) of the Hardwires/Buses tab. Associations made in the hardwires interface are reflected here.

- Enable the **Reserved for Routing** checkbox if the channel should be marked as reserved for routing.
- Click the **Disabled** checkbox to disable the channel.

Disabled channels are not used in connections/routes. Disabling a channel can be useful for making unused channels or channels with a damaged relay inactive.



- Click  **Save** on the NI Switch Executive taskbar to save the changes.



Tip By using filters and selecting multiple items, you can perform mass configurations. Multiselection is particularly helpful when defining hardwires and configuration channels.

Adding Mutual Exclusions

Complete the following steps to add a mutual exclusion:

- Launch NI MAX with NI Switch Executive.
- Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
- Select the NI Switch Executive virtual device to modify.
- Click the **Channels/Exclusions** tab.
- Click  at the top of the **Exclusions** listbox to add a mutual exclusion.
- Type a name for the exclusion in the **Name** textbox or accept the predefined name.
- Select the channels to exclude from the **Available Channels** listbox and click  to add each channel to the **Mutually Excluded Channels** listbox.
- Type any comments you have in the **Comment** textbox.



Note If needed, select the **Disabled** checkbox to disable the exclusion.

- Click on the NI Switch Executive taskbar to save the changes.

Adding Set Exclusions

Complete the following steps to add a set exclusion:

- Launch NI MAX with NI Switch Executive.
- Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
- Select the NI Switch Executive virtual device to modify.
- Click the **Channels/Exclusions** tab.
- Click at the top of the **Exclusions** listbox to add a set exclusion.
- Type a name for the exclusion in the **Name** textbox or accept the predefined name.
- Select the channels to exclude from the **Available Channels** listbox and move them to the **Set 1 Channels** listbox or **Set 2 Channels** listbox by clicking and .



Note If you cannot see enough information in each listbox column to make your choices, try hiding the MAX configuration tree to create more space in the user interface.

- Type any comments you have in the **Comment** textbox.
- Click on the NI Switch Executive taskbar to save the changes.

Deleting Exclusions

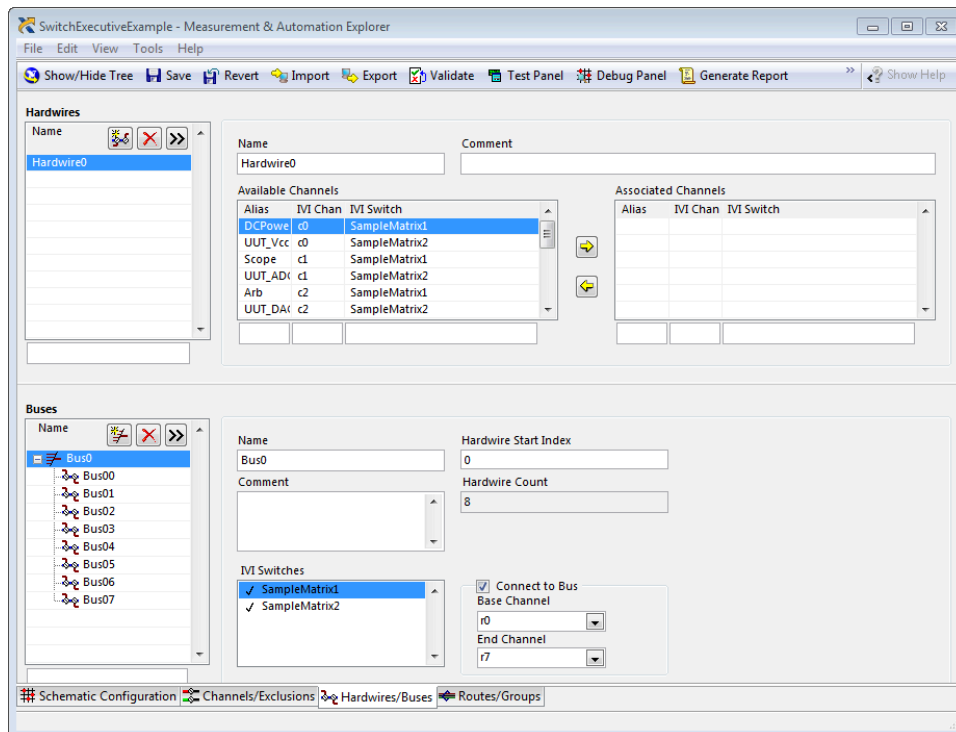
Complete the following steps to delete an exclusion:

- Launch NI MAX with NI Switch Executive.
- Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
- Select the NI Switch Executive virtual device to modify.
- Click the **Channels/Exclusions** tab.
- Select the exclusion you want to delete from the **Exclusions** listbox.
- Click . A dialog box opens to confirm the deletion.
- Click **Yes** to delete the exclusion.
- Click on the NI Switch Executive taskbar to save the changes.

Hardwires/Buses Tab


Use the **Hardwires/Buses** tab to configure hardwires and buses in NI Switch Executive.



Hardwires that belong to a bus only display in the **Buses** listbox and are nested underneath their respective bus. Hardwires that do not belong to a bus display in the **Hardwires** listbox.



Adding Hardwires to Define Multidevice Topologies

Complete the following steps to define a hardware:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify.
4. Click the **Hardwires/Buses** tab.
5. Click  at the top of the **Hardwires** listbox to add a hardware.

6. Type a name for the hardwire in the **Name** textbox or the accept the predefined alias of the hardwire.
7. Select the channels that are associated with the hardwire in the **Available Channels** listbox and move them to the **Associated Channels** listbox by clicking 
8. Type any comments you have in the **Comment** textbox.
9. Click  on the NI Switch Executive taskbar to save the changes.



To continue configuring your NI Switch Executive virtual device, refer to [Adding Route Groups](#).

Deleting Hardwires




Note Deleting hardwires can break routes that require the hardwire. Always [validate](#) your NI Switch Executive system after you delete hardwires.

Complete the following steps to delete a hardwire:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify.
4. Click the **Hardwires/Buses** tab.
5. Select the hardwire you want to delete in the **Hardwires** listbox.
6. Click . A dialog box opens to confirm the deletion.
7. Click **Yes** to delete the hardwire.
8. Click  on the NI Switch Executive taskbar to save the changes.

Adding Buses


Complete the following steps to add a bus:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify.
4. Click the **Hardwires/Buses** tab.
5. Click  at the top of the **Buses** listbox to add a bus.

6. Type a name for the bus in the **Name** textbox or accept the predefined alias of the bus.
7. Select a switch to associate with the bus from the **IVI Switches** listbox.
8. Enable the **Connect to Bus** checkbox.
9. Select a channel from the **Base Channel** pull-down menu. The base channel is the first channel of the series of hardwires that are created.
10. Select the last channel in the series of created hardwires from the **End Channel** pull-down menu.



Note Some switch devices do not support creating multi-channel buses. If the switch device you selected from the **IVI Switches** listbox does not support multi-channel bus creation, you will only be able to select the same channel from the **End Channel** pull-down menu as the channel selected in the **Base Channel** pull-down menu.

11. Accept the predefined **Hardwire Start Index** value or type a new number in the **Hardwire Start Index** textbox. This value is the first number appended to the end of the generated hardwires. For example, if the start index is 2, `BusName2`, `BusName3`, `BusName4`, and so on are the default names created, where `BusName` is the name of the newly created bus.
12. Select another switch from the **IVI Switches** listbox to associate with the bus.
13. Enable the **Connect to Bus** checkbox.
14. Select a channel from the **Base Channel** pull-down menu.
15. Select the last channel in the series of created hardwires from the **End Channel** pull-down menu.
16. Accept the predefined **Hardwire Start Index** value or type a new number in the **Hardwire Start Index** textbox.
17. Type any comments you have in the **Comment** textbox.
18. Repeat steps 12 through 16 for any additional switches associated with the bus.
19. Click  on the NI Switch Executive taskbar to save the changes.



Note Buses are useful for simplifying sequential associations. If you have non-sequential channels, you can use individual [hardwires](#) to configure the NI Switch Executive virtual device.

Converting Buses to Hardwires

For additional granularity, a bus can be converted to individual hardwires.

Complete the following steps to convert a bus to individual hardwires.





Caution After a bus has been converted to individual hardwires, the bus cannot be recreated from these hardwires.

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify.
4. Click the **Hardwires/Buses** tab.
5. Right-click the bus you want to convert to individual hardwires from the **Buses** listbox and select **Convert Bus to Hardwires**.
6. Click **Yes** to complete converting the bus to individual hardwires.

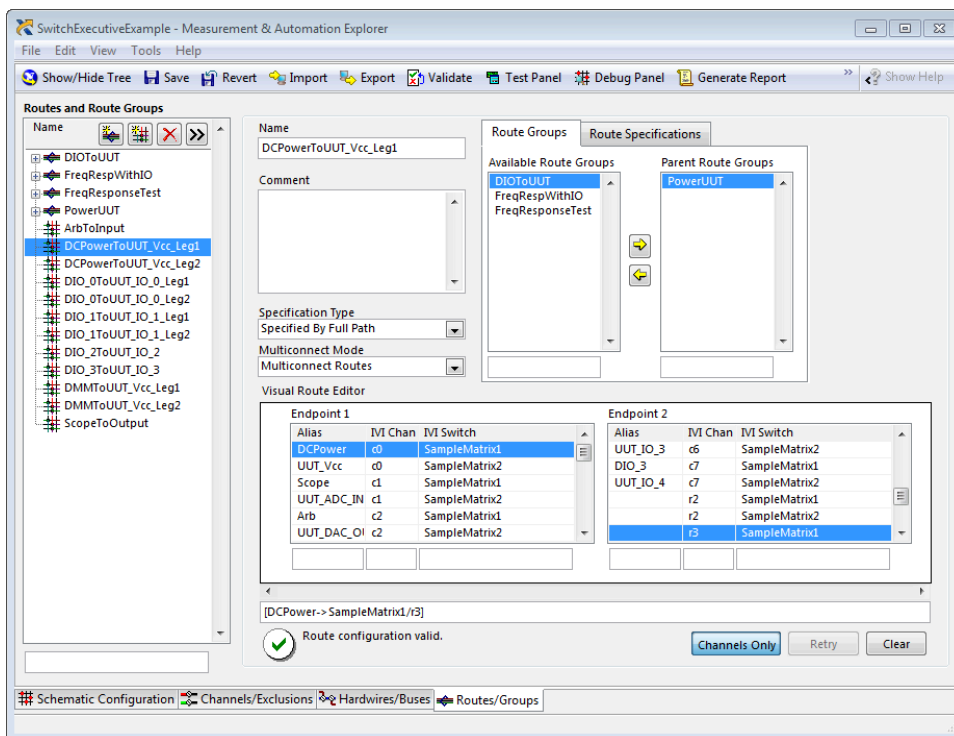
Deleting Buses

Complete the following steps to delete a bus:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device to modify.
4. Click the **Hardwires/Buses** tab.
5. Select the bus you want to delete from the **Buses** listbox.
6. Click . A dialog box opens to confirm the deletion.
7. Click **Yes** to delete the bus.
8. Click  on the NI Switch Executive taskbar to save the changes.



Routes/Groups Tab

Use the **Routes/Groups** tab to configure routes and route groups in NI Switch Executive.



Adding Routes

Complete the following steps to add a route:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device that you want to modify.
4. Click the **Routes/Groups** tab.
5. Click  to add a new route. A new route appears selected in the list of routes.
6. Type a name for the route in the **Name** textbox or accept the predefined alias of the route.
7. Type any comments you may have in the **Comment** field.
8. Select the default multiconnect behavior intended for the route in the **Multiconnect Mode** drop-down listbox:
 - **Multiconnect Routes**—Allows NI Switch Executive to use this route between multiple calls.
 - **No Multiconnect**—Prevents sharing between multiple calls.
9. Click the **Route Groups** tab.
10. Select the route group(s) that you want to associate with the route from the **Available Route Groups** listbox and click  to move the route groups to the

Parent Route Groups list.



Tip You can also associate a route with a route group by dragging the route and dropping it on top of the route group in the **Routes and Route Groups** listbox.

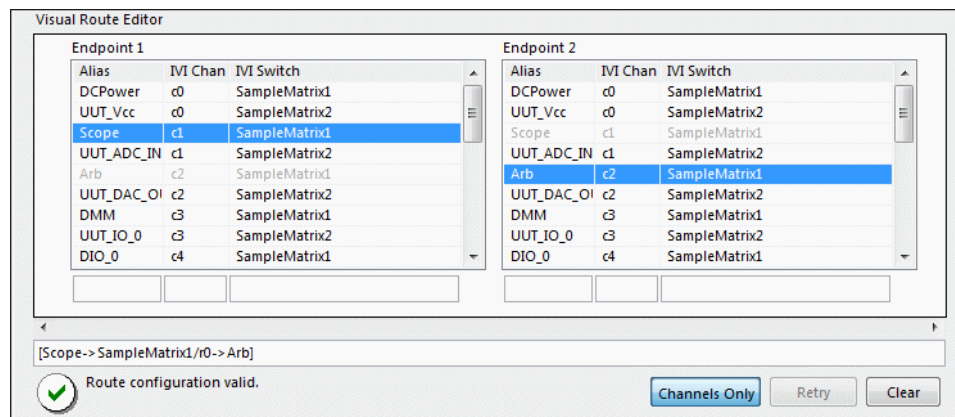
Route groups help NI Switch Executive identify unintentional cross-connects. For example, if you are creating a test system, it is likely that you are using multiple routes at the same time. To ensure that these routes do not cross paths unintentionally, they must be dependent on one another. Use groups to prevent future errors when using your NI Switch Executive virtual device.

- (Optional) Click the **Route Specifications** tab and set route constraints. The values in this tab filter endpoints in the Visual Route Editor window based on the specifications of the route and/or the wiring mode of the switch module.



Tip You can use scientific units in this tab. For example, instead of entering 1000000, you can enter 1M, or you can enter 1m for 0.001.



- Complete the following steps to define endpoints—or the beginning and end of the route path—using the two tables in the **Visual Route Editor** window as shown in the following figure:



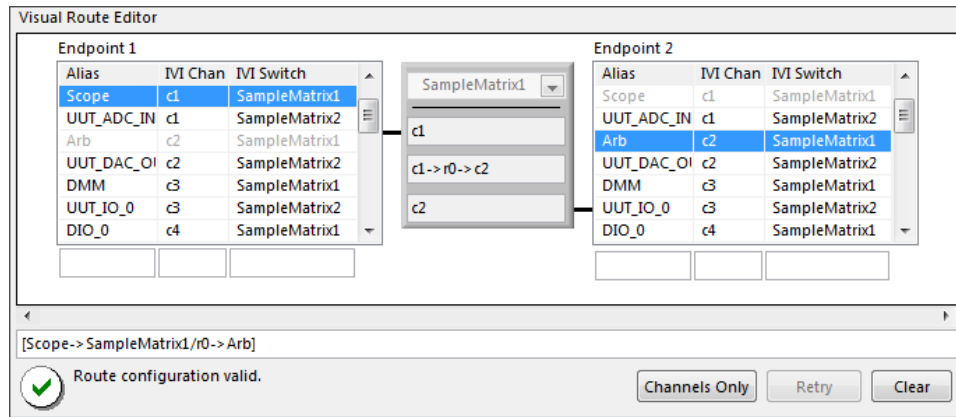
- Select the first endpoint of the route in the **Endpoint 1** table.
- Select the second endpoint of the route in the **Endpoint 2** table. The [path editor](#) automatically populates with the route specification string that designates your route path.




Tip Experiment with route selections to see how your changes affect the route specification string of a route path.

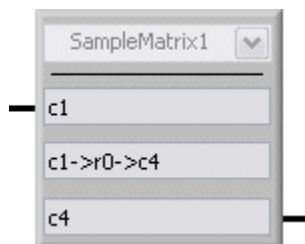
Status icons,  and , below the path editor indicate if there are any errors or warnings or if the path is valid.

- (Optional) Click **Channels Only** to enable visibility of the switch box between the Endpoint 1 and Endpoint 2 tables, as show in the following figure:

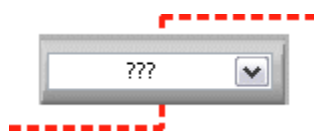


 **Note** By default, the switch box does not display between the Endpoint 1 and Endpoint 2 tables. If you do not see the switch box, click **Channels Only**.

If the two endpoints can be connected, a solid black line displays between the two selected endpoints, indicating that the path is complete.



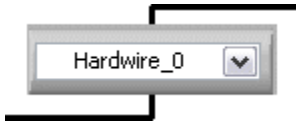
If the route path is invalid, the lines between the channels display as dashed red lines.



If you want to modify the path selected, you can use the [path editor](#) to make your changes.

- (Optional) Define hardwires to complete the route.

- If the two endpoints are in separate switch modules physically connected by a hardwire, use the drop-down listbox that opens between the two endpoint tables to select another [hardwire](#) to complete the route. You can use the drop-down listbox to manually select different routing channels and hardwires.




Note The drop-down listbox will only display if the switch box has been enabled by clicking **Channels Only**.

- If the two endpoints of a route are in separate switch modules connected by one or more intermediate switches, you must define the hardwires that connect them.

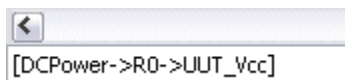


Note Hardwires are defined by a user. For information about defining new hardwires, refer to [Adding Hardwires to Define Multidevice Topologies](#).

- (Optional) Click **Retry** to attempt rerouting if you made changes to your virtual device configuration that might cause a route to become invalid, for example, adding/removing hardwires or exclusions.
- Click  **Save** on the NI Switch Executive taskbar to save the changes.

Adding Routes with the Path Editor

Another method of creating or modifying a route is to use the path editor below the **Visual Route Editor** window.





The path editor is automatically populated with the route specification string based on the selections you have made in the Visual Route Editor window.

To use the path editor, type the route path in the path editor textbox in the appropriate format, and press <Enter> to apply your changes. Click **Save** to save your changes, or click **Clear** to cancel your changes. Refer to [Route Specification Strings](#) for more

information.


Deleting Routes


Complete the following steps to delete a route:


1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device that you want to modify.
4. Click the **Routes/Groups** tab.
5. Select the route you want to delete from **Routes and Route Groups** listbox.
6. Click . A dialog box opens to confirm the deletion.
7. Click **OK** to delete the route.
8. Click  on the NI Switch Executive taskbar to save the changes.

Adding Route Groups

Complete the following steps to add a route group:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device that you want to modify.
4. Click the **Routes/Groups** tab.
5. Click  to create a route group.
6. Type a name for the route group in the **Name** textbox or accept the predefined alias of the route group.
7. Drag and drop the routes and/or route groups in the **Routes and Route Groups** listbox onto the route group to populate the route group.



Another way to add routes and/or route groups is through the **Available Routes and Route Groups** listbox. Select the routes and/or route groups you want to associate with the route group you created and move them to the **Child Routes and Route Groups** listbox by clicking .

8. Type any comments you have in the **Comment** textbox.
9. Click  on the NI Switch Executive taskbar to save the changes.

To continue configuring your NI Switch Executive virtual device, refer to [Adding Routes](#).


Deleting Route Groups

Complete the following steps to delete a route group:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the NI Switch Executive virtual device that you want to modify.
4. Click the **Routes/Groups** tab.
5. Select the route group you want to delete from the **Routes and Route Groups** listbox.
6. Click . A dialog box opens to confirm the deletion.
7. Click **OK** to delete the route group.
8. Click  on the NI Switch Executive taskbar to save the changes.

Validating a Virtual Device

Complete the following steps to validate an NI Switch Executive virtual device:


1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the virtual device you want to validate.
4. Click  on the NI Switch Executive taskbar in MAX. A dialog box opens.
5. (Optional) Select the **Use Simulation Mode** checkbox to simulate connections during validation.



Note Simulation mode simulates the configuration of your virtual device, performs a valid system check, and reduces the wear and tear on switch relays. Simulation mode does not physically communicate with your switches.





Caution If you do not use Simulation mode, ensure that you have disconnected any fixturing or wiring that might be damaged by connecting all of your routes and route groups.

6. Click the checkbox items you want to validate: **IVI Configuration**, **Routes**, and/or **Route Groups**.
7. Click **Next**. A window displays the validation status of the items in your configuration and if the validation passes or fails.
8. Proceed to step 9 if the validation test passes. If the validation status includes any errors, correct the configuration and repeat steps 4 through 7.
9. Click **Finish**. Now you can program and/or deploy the virtual device.
10. Click  on the NI Switch Executive taskbar to save the configuration.

Generating a Report

Complete the following steps to generate a report of your NI Switch Executive virtual device configuration:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Select the virtual device you want to report.
4. Click  **Generate Report**. A file window opens.
5. Select a location to save your report and click **OK**.
6. Click  on the NI Switch Executive taskbar to save the changes.

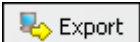
The report is generated in HTML.

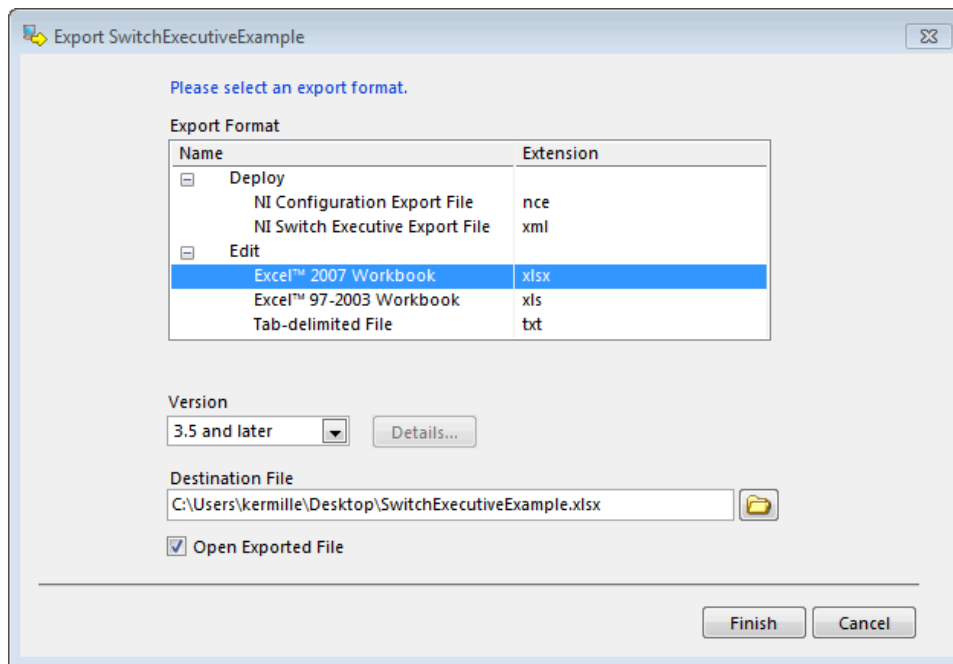


Tip The report is formatted in such a way that it can be easily imported into many popular spreadsheet software packages.

Exporting a Virtual Device

Complete the following steps to export the configuration file of an NI Switch Executive virtual device:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Right-click a virtual device and select  **Export**. The Export dialog box opens.



4. Select an Export Format for the virtual device configuration. NI Switch Executive supports the following export and import formats:
 - **NI Configuration Export File (. nce)**—Recommended for export and import of complete configurations that do not need editing outside of NI Switch Executive. Allows saving of system configuration information for all devices and interfaces, including NI Switch Executive virtual devices.
 - **NI Switch Executive Export File (. xml)**—Recommended for export and import of individual virtual device configurations that do not need editing outside of NI Switch Executive.



Caution When exporting a virtual device configuration that contains non-ASCII characters (0X000-0X07F) (e.g. ä, é, ó, ñ) to XML, NI Switch Executive extracts those special characters, and replaces them with blank spaces.

- **Microsoft Excel Workbooks (. xls and . xlsx)**—Recommended for export and import of individual virtual device configurations in a format suitable for editing. Components of the switching system (virtual devices, channels, routes, and so on) are contained in separate workbook sheets.
- **Tab-delimited File (. txt)**—Recommended for export and import of individual virtual device configurations in a format suitable for editing. When opened in a spreadsheet editing program, tab-delimited files contain the components of the switching system (virtual devices, channels, routes, and so on) in a single workbook sheet.

5. Select the NI Switch Executive version to export to from the **Version** drop-down listbox. Click the **Details** button to show what data will be left out for older versions of NI Switch Executive. The default value for the version is the current (latest) version. Change this value if you want to export to an older version.
6. Specify a Destination File for the virtual device configuration.

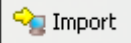


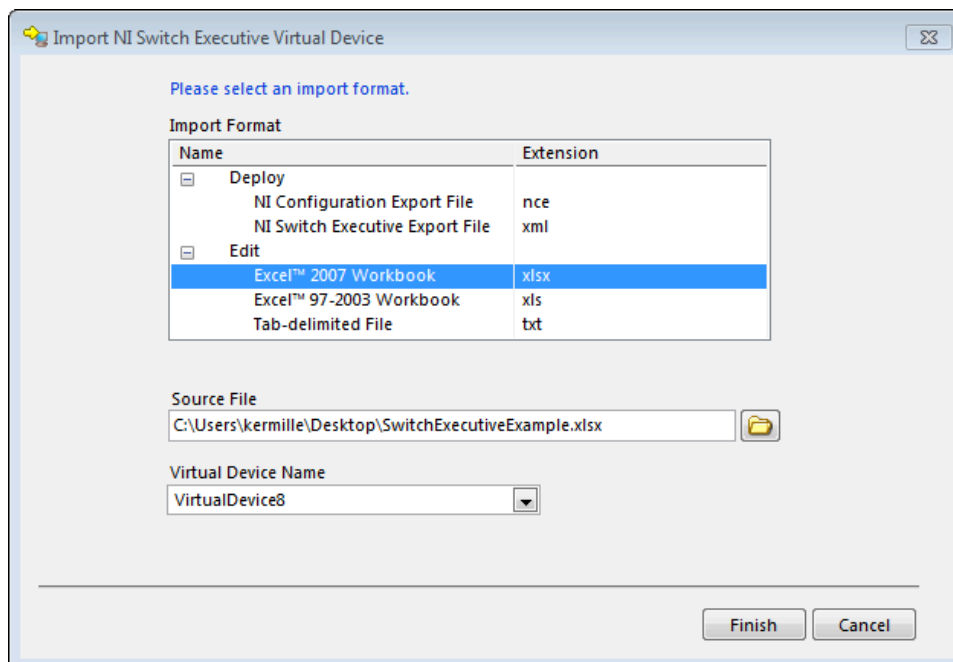
Caution Exporting overwrites files with the same filename. To avoid the loss of custom data in modified files, enter a unique Destination File for each export.

7. Click **Finish** to export the virtual device configuration. By default, the file is opened after export.

Importing a Virtual Device

Complete the following steps to import the configuration file of an NI Switch Executive virtual device:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces**.
3. Right-click **NI Switch Executive Virtual Devices** and select . The Import dialog box opens.



4. Select an **Import Format** for the virtual device configuration. NI Switch Executive

supports the following import formats.



Note If using the NI Configuration Export (.nce) file format, skip to step 7.

- **NI Configuration Export File (.nce)**—Recommended for import and export of complete configurations that do not need editing outside of NI Switch Executive. Allows saving of system configuration information for all devices and interfaces, including NI Switch Executive virtual devices.
 - **NI Switch Executive Export File (.xml)**—Recommended for import and export of individual virtual device configurations that do not need editing outside of NI Switch Executive.
 - **Microsoft Excel Workbooks (.xls and .xlsx)**—Recommended for import and export of individual virtual device configurations in a format suitable for editing. Components of the switching system (virtual devices, channels, routes, and so on) are contained in separate workbook sheets.
 - **Tab-delimited File (.txt)**—Recommended for import and export of individual virtual device configurations in a format suitable for editing. When opened in a spreadsheet editing program, tab-delimited files contain the components of the switching system (virtual devices, channels, routes, and so on) in a single workbook sheet.
5. Select the **Source File** to import. The source file is the file that contains the virtual device configuration you exported from the development system.
 6. Select a name for the virtual device from the **Virtual Device Name** drop-down listbox.



Note To ensure that Virtual Device Name matches the virtual device name in the file, NI recommends that you always select (**Extract name from source file**).



Note Make sure the NI Switch Executive virtual device name matches that used in your test code.

7. Click **Finish** to import the virtual device configuration. The NI Switch Executive virtual device configuration is loaded into MAX.



Note (XML file format only) If any errors occur during this process (such as an IVI device not being accessible), the virtual device is created, but the IVI switches that caused errors are not added. When the IVI switches are not added, the channels of those switches are

also not added, which can break routes or route groups. You can [manually add](#) the IVI switches later.

Deploying a Virtual Device

Complete the following steps to deploy an NI Switch Executive virtual device from a development system to a target system.



Note A development system must be running a [licensed](#) development version of NI Switch Executive. Target, or deployment, systems can run either a licensed deployment or a licensed development copy of NI Switch Executive.

1. [Export](#) the configuration files from the development system with the same NI Switch Executive version as the target system.
2. [Import](#) the configuration files to the target system.
3. [Validate](#) the deployment of the NI Switch Executive virtual device.

Exporting Configurations

Complete the following steps on the development system to export the configuration files necessary to deploy an NI Switch Executive virtual device:

1. [Export](#) the NI Switch Executive virtual device configuration file. If you are exporting a virtual device to a `.nce` file, skip steps 2 and 3.
2. Export the DAQmx configuration file if the NI Switch Executive virtual device you are deploying uses DAQmx switches.
 1. Launch Measurement & Automation Explorer (MAX).
 2. Export the DAQmx configuration file using the Export Configuration Wizard in MAX. Refer to the ***Importing and Exporting Configurations*** topic of the ***Measurement & Automation Explorer Help for NI-DAQmx*** for more information by navigating in MAX to **Help»Help Topics»NI-DAQmx»MAX Help for NI-DAQmx**.
3. [\(Optional\) Copy the IVI configuration file from the development system to the target system if you are using IVI devices.](#)
 - (IVI Compliance Package 3.2 or higher) `IVIConfigurationStore.xml`, the IVI configuration file, is installed at `Documents and Settings\All`

Users\Application Data\IVI Foundation\IVI.

- (IVI Compliance Package 2.0 to 3.1) IVIConfigurationStore.xml, the IVI configuration file, is installed at Program Files\IVI\Data.
- (IVI Compliance Package 1.x) IVI.ini, the IVI configuration file, is installed at VXIPNP\WINNT[WIN95]\niivi.



Note The IVI configuration file contains IVI Driver Sessions and IVI Logical Names. Because IVI Driver Sessions and IVI Logical Names must match between the development and target deployment systems, copy this file to the exact directory location on the target system as where it was installed in the development system.

Importing Configurations

After you [export](#) configuration files from the development system, complete the following steps on the target system to import the configuration files necessary to deploy an NI Switch Executive virtual device:

1. Verify that the target has the following items installed:
 1. A [licensed](#) deployment or development copy of NI Switch Executive
 2. Windows 8.1/8.0/7/Vista/XP SP3
 3. IVI instrument drivers for your switch modules



Note NI currently supports C class drivers only. You can create your own C class IVI drivers with Measurement Studio templates—specifically in LabWindows/CVI. For more information about Measurement Studio, visit ni.com.

4. Skip steps e and f if you are importing a virtual device from a .nce file.
5. If the virtual device you are deploying uses DAQmx switches, a DAQmx configuration file imported from the development system from which you are deploying. Complete the following steps to import the DAQmx configuration file:
 1. Launch Measurement & Automation Explorer (MAX).
 2. Import the DAQmx configuration file using the Import Configuration Wizard in MAX. Refer to the ***Importing and Exporting Configurations*** topic of the ***Measurement & Automation Explorer Help for NI-DAQmx*** for more information by navigating in MAX to **Help»Help Topics»NI-DAQmx»MAX Help for NI-DAQmx**.

6. (Optional) If you are using IVI devices, an [IVI configuration file](#) copied from and installed in the same directory as the development system from which you are deploying. Copying the IVI configuration file from the development system to the target system ensures that both systems are using the same IVI Logical Names.
2. [Import](#) the NI Switch Executive virtual device configuration file.

Validating Deployment

[Validate](#) the deployment of an NI Switch Executive virtual device to ensure the following:

- The correct IVI switches are present and named properly
- Routes are valid
- Route groups are valid

Deleting a Virtual Device

Complete the following steps to delete an NI Switch Executive virtual device:

1. Launch NI MAX with NI Switch Executive.
2. Expand **Devices and Interfaces»NI Switch Executive Virtual Devices**.
3. Right-click the NI Switch Executive virtual device you want to delete and select **Delete**. A dialog box opens to confirm the deletion.



Tip To delete multiple virtual devices at one time, hold <Shift> or <Ctrl> while selecting multiple virtual devices and press <Delete>.

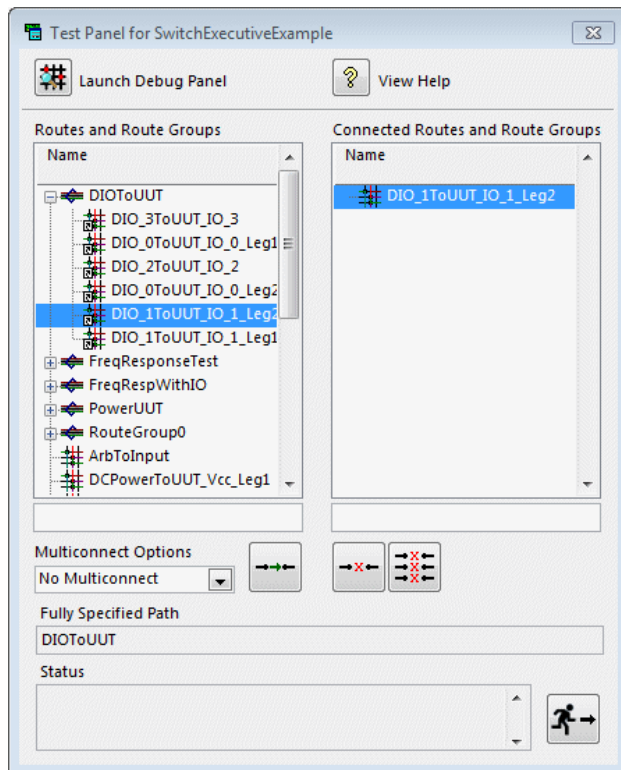
4. Click **Yes** to delete the virtual device or **No** to abort.



Note When you create a new virtual device, NI Switch Executive automatically creates default IVI configurations for **all** NI switch modules detected in your system. If you remove an NI switch module from **Devices and Interfaces** or replace it with a different device, you should also remove the corresponding IVI configuration items from **IVI Drivers**, such as Logical Names.

NI Switch Executive Test Panel

Use the NI Switch Executive Test Panel to test the individual routes and route groups associated with a virtual device. When managing or developing applications, use the Test Panel to individually connect and disconnect selected routes or route groups and verify that they are functioning correctly.



Accessing the Test Panel

You can access the NI Switch Executive Test Panel from within NI MAX with NI Switch Executive one of two ways:

- Right-click a virtual device and select **Test Panel**.

or

- Select a virtual device and click the **Test Panel** button on the taskbar.

Using the Test Panel

	Launch Debug Panel —Launches the NI Switch Executive Debug Panel. Use the debug panel to monitor the state of your live NI Switch Executive session.
	View Help —Displays information about the Test Panel.

Routes and Route Groups—Displays all the routes and route groups associated with an NI Switch Executive virtual device, including shortcuts to routes and routes groups in parent route groups. Routes, followed by route groups, child routes, and child route groups, are displayed in alphabetical order.

Select a route or route group and click the **Connect Selected Route or Route Group** button to connect the route or route group.

Connected Routes and Route Groups—Displays all the routes and route groups connected on an NI Switch Executive virtual device in connection order (earliest to most recent connection). Child route groups are **not** displayed in the **Connected Routes and Route Groups** listbox.

Select a route or route group and click the **Disconnect Selected Route or Route Group** button to disconnect the route or route group. When a route or route group is disconnected, it is removed from the **Connected Routes and Route Groups** listbox. To disconnect all routes and clear the listbox, click the **Disconnect All Routes and Route Groups** button.

Filter TextBoxes—Use the [filter textboxes](#), located directly below the route and route groups listboxes, for faster searches. In the filter textbox that corresponds to the listbox you want to search, type in the first characters of the route or route group. All names that match the search string display in the listbox. The search string is **not** case sensitive.

	Connect Selected Route or Route Group —Connects the selected route or route group. If the connection is successful, the route or route group is added to the Connected Routes and Route Groups listbox. If an error occurs, an error description displays in the Status indicator.
	Disconnect Selected Route or Route Group —Disconnects the selected route or route group and removes it from the Connected Routes and Route Groups listbox. If an error occurs, an error description displays in the Status indicator.



Disconnect All Routes and Route Groups—Disconnects all connected routes and route groups and removes them from the **Connected Routes and Route Groups** listbox.

Multiconnect Options—Select one of the following [multiconnect options](#) for the connection operation:

- **Use Default**—Uses the configured multiconnect options for the route or route group.
- **Multiconnect Routes**—Connects the selected route or route group multiple times. Each connection is displayed in the **Connected Routes and Route Groups** listbox.
- **No Multiconnect**—Connects the selected route or route group only once. Because it is connected only once, the route or route group can be disconnected only once. If you attempt to connect the same route or route group multiple times, an error displays in the Status indicator.

Fully Specified Path—Displays the fully specified path, including all endpoints, for a selected route or route shortcut.



Note For a selected route group or route group shortcut, the name of the route group displays in Fully Specified Path.

Status—Displays the status of the last operation.



Exit—Closes the NI Switch Executive Test Panel.

NI Switch Executive Debug Panel

The NI Switch Executive Debug Panel allows you to monitor the state of your live NI Switch Executive session. You can use the Debug Panel to debug your application by looking at your connection states and action log.

Live NI Switch Executive sessions include the following sources:

- Virtual device validation
- Route and route group testing in the Test Panel
- A running application that uses the NI Switch Executive API

For more information about the Debug Panel, refer to the following topics:

[Accessing the NI Switch Executive Debug Panel](#)

[Creating a New Debug Panel Window](#)

[Debug Panel State and Action Log Windows](#)

[Customizing the Debug Panel](#)

[Debug Panel Toolbar](#)

[Debug Panel Keyboard Shortcuts](#)

Accessing the NI Switch Executive Debug Panel

You can access the NI Switch Executive Debug Panel from within NI MAX with NI Switch Executive or from the **Start** menu.

From MAX

- Right-click a virtual device and select **Debug Panel**.

or

- Click  **Debug Panel** in the NI Switch Executive taskbar.

From the Start Menu

Select **Start»All Programs»National Instruments»Switch Executive»NI Switch Executive Debug Panel**.

Creating a New Debug Panel Window

Create a new window to monitor multiple NI Switch Executive sessions simultaneously. You can create a new window for the same device or different devices without causing changes in the hardware.

Select **File»New Window** or press <Ctrl-N> to open multiple instances of the NI Switch

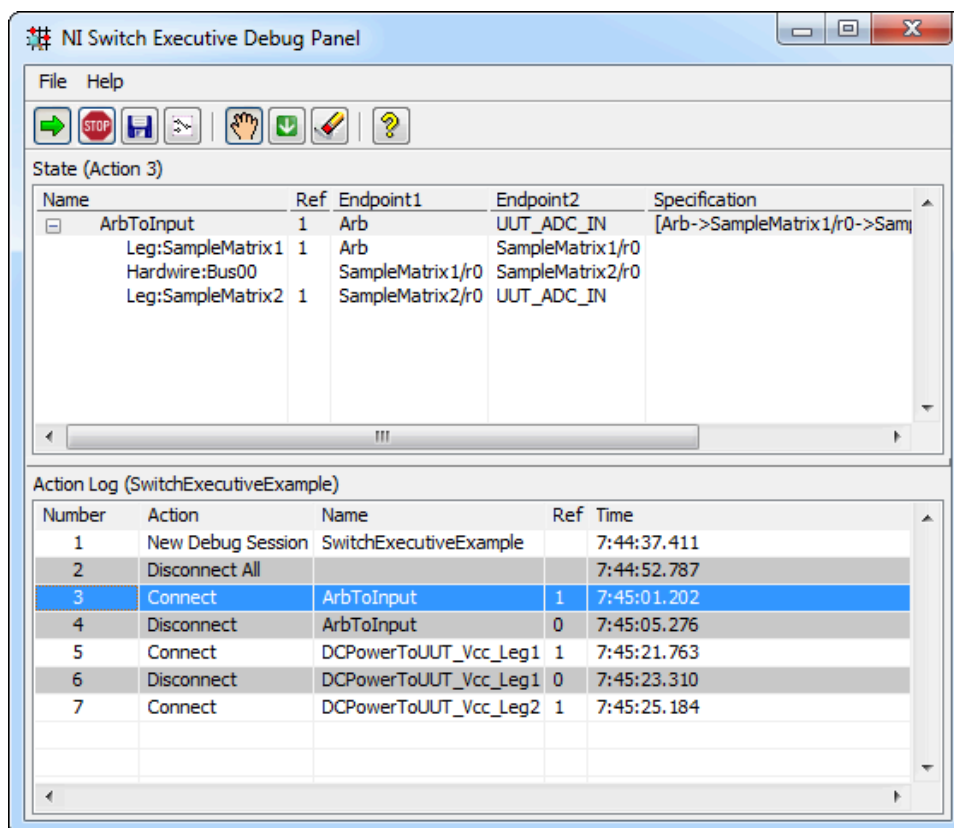
Executive Debug Panel.

Debug Panel State and Action Log Windows

The NI Switch Executive Debug Panel is divided into **State** and **Action Log** windows.

The **State** and **Action Log** windows display live NI Switch Executive session information from the following sources:

- Virtual device validation
- Route and route group testing in the Test Panel
- A running application that uses the NI Switch Executive API



State Window

The **State** window displays the latest state of the session if the last action in the **Action Log** window is selected or a prior state in an earlier action is selected. The **State** window contains the following columns:

- **Name**—Name of route, route group, or path.
- **Ref**—Reference count for the route, route group, or path. The number of times connected.
- **Endpoint1**—Channel name for first endpoint of route or path. The channel display style dictates the style of this column.
- **Endpoint2**—Channel name for second endpoint of route or path. The channel display style dictates the style of this column.
- **Specification**—Full path string of the route, route group, or path.

Action Log Window

The **Action Log** window displays the actions of the session in chronological order. The **Action Log** window contains the following columns:

- **Number**—Number of action ascending in chronological order.
- **Action**—Name of action, or connection change.
- **Name**—Name for action target, such as name of route or name of virtual device.
- **Ref**—Reference count for the route, route group, or path. The number of times connected.
- **Time**—Time stamp for action.









Customizing the Debug Panel

Complete the following steps to customize the debug panel:

1. Choose **File»Options** to customize the following:
 - How channel names display using the **Channel Names** option buttons
 - How actions are logged using the **Action Log Depth (maximum lines)** textbox
2. Enable the **Synchronous Mode** checkbox in the following instances:
 - If you are debugging a C or C++ application and want actions properly logged between breakpoints set in the application
 - If you want to ensure an action is transferred to any panels before a breakpoint breaks the application

The NI Switch Executive Debug Panel Toolbar

Use the buttons in the debug panel toolbar when debugging your application.

Button	Function
	Click the Start button to start capture of actions and states.
	Click the Stop button to stop capture of actions and states.
	Click the Save button to save the state and/or action log to a <code>.log</code> file.
	Click the Launch NI-SWITCH Soft Front Panel button to launch the NI-SWITCH Soft Front Panel.
	Click the Autoscroll Actions button to toggle autoscroll of the actions list. Enable autoscroll to automatically show the latest state with each new action of the session.
	Click the Jump to Last Action button to select the last action in the action log window.
	Click the Clear Actions button to remove all actions from the action log window.
	Click the Show Help button to display the Debug Panel section of the NI Switch Executive Help .

Debug Panel Keyboard Shortcuts

The following table lists keyboard shortcuts you can use with the NI Switch Executive Debug Panel:

Shortcut	Action
<Ctrl-F1>	Displays online help
<Ctrl-N>	Creates a new window
<Ctrl-S>	Launches the Save dialog box
<Ctrl-O>	Launches the Options dialog box
<Ctrl-Q>	Exits the debug panel

Concepts

Expand this topic to view the conceptual information related to using an NI Switch Executive virtual device.

Configuration

Configure a virtual device to create new definitions and modify existing configuration data of the switches, channels, hardwires, buses, exclusions, routes, and route groups that compose a virtual device.

Some examples of configuration tasks include:

- Adding or removing an IVI switch from a virtual device
- Configuring channels
- Defining an exclusion to create a connection rule
- Adding hardwires and/or a bus to define multidevice topologies
- Adding routes
- Adding route groups

Related Topics

[Configuring a Virtual Device \(in MAX\) Using the Configuration API](#)

Validation

Validate a virtual device to ensure that IVI configurations, routes, and route groups are valid and route group members do not use conflicting resources.

You should always validate after making any modifications to the virtual device configuration and after deploying a virtual device.

Related Topics

[Validating a Virtual Device \(in MAX\) Using the Configuration API NI Switch Executive Test Panel](#)

Programming

Program a virtual device using an application development environment (ADE) such as LabVIEW, NI TestStand, LabWindows/CVI, or Visual Basic. Because an ADE recognizes a virtual device as a single instrument, programming time is significantly reduced.

Related Topics

[Programming with NI Switch Executive](#)

Export

Export the configuration of a virtual device to perform configuration management tasks in other file formats or with other programs such as Microsoft Excel.

Related Topics

[Exporting a Virtual Device \(in MAX\) Using the Configuration API](#) [Managing a Virtual Device in Excel](#)

Import

After you have managed the configuration of a virtual device in other file formats or with other programs such as Microsoft Excel, import the configuration back to NI Switch Executive to continue working with the virtual device.

Related Topics

[Importing a Virtual Device \(in MAX\) Using the Configuration API](#)

Deployment

Deploy a virtual device you developed on one computer for use on one or more other computers.



Note A development system must be running a [licensed](#) development version of NI Switch Executive. Target, or deployment, systems can run either a licensed deployment or a licensed development copy of NI Switch Executive.

Related Topics

[Deploying a Virtual Device \(in MAX\) Using the Configuration API](#)

Report Generation

Generate a detailed report that includes the definitions and configurations of the IVI switch modules, channels, hardwires, buses, exclusions, routes, and route groups that compose a virtual device.

Related Topics

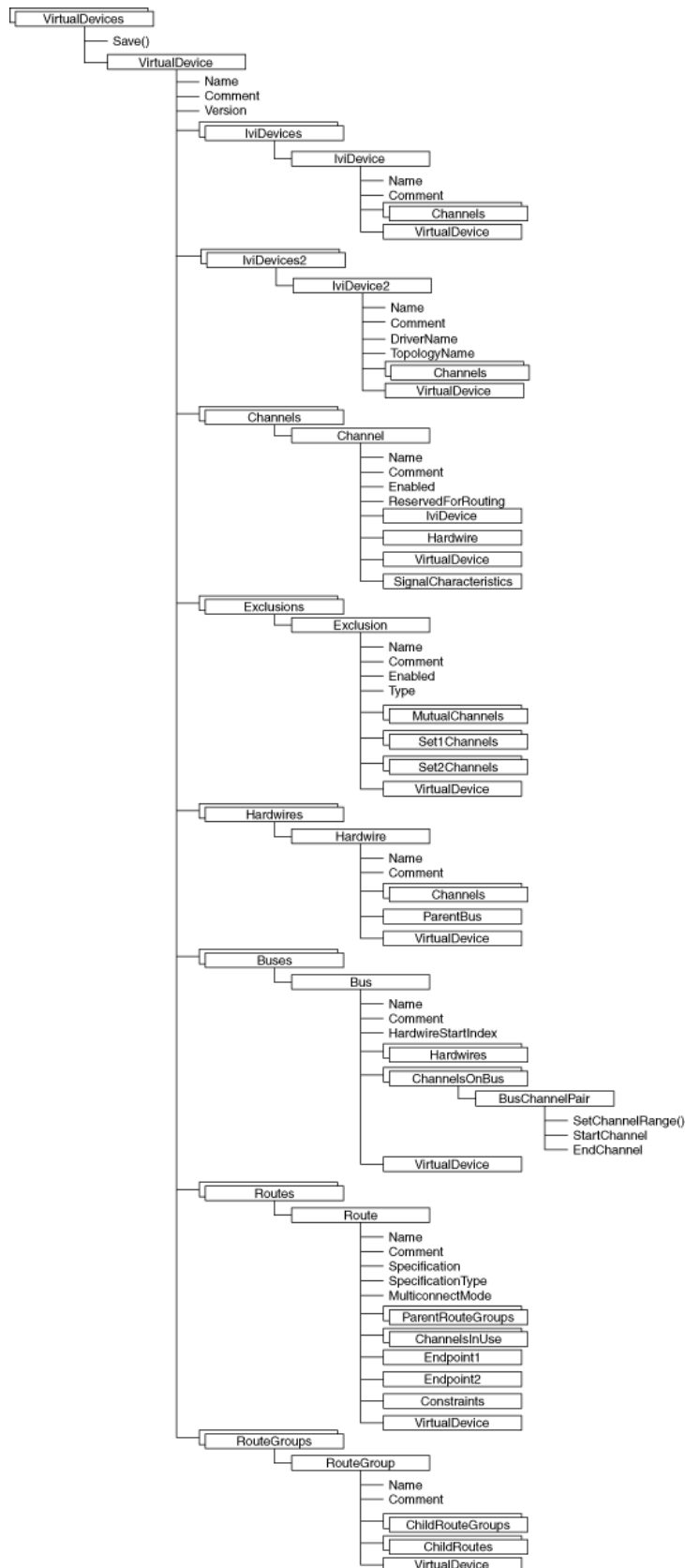
[Generating a Report \(in MAX\) Using the Configuration API](#)

Using the Configuration API

The configuration API provides a programmatic interface to configure your NI Switch Executive virtual devices. Use the configuration API for the following tasks:

- **Virtual device creation/editing**—Create new virtual devices programmatically and edit existing configurations. The configuration API provides programmatic access to IVI devices, channels, exclusions, hardwires, buses, routes, and route groups.
- **Configuration management**—Programmatically import, export, delete, and rename NI Switch Executive configurations.
- **Report generation**—Create both simple and advanced reports by reading the configuration associated with a specified virtual device.

Object Hierarchy



Additional Considerations

Differences exist between creating and editing your configuration in MAX and doing so programmatically:

- In MAX, you can change the hardware on a channel from the channel details. Programmatically, you can only read the hardware on the channel. To change the hardware on a channel programmatically, you must first disassociate that channel from its current hardware (if it was on a hardware), and then associate it with another hardware.
- The MAX UI configuration performs extensive checks when creating routes. When programmatically creating routes and route groups, NI Switch Executive performs only a portion of these checks:
 - You can programmatically set the constraints for a route, but these constraints are not checked when you set the route specification.
 - You can programmatically create a route that uses channels with different wire modes.
 - You can programmatically set the route specification to any well-formed string of channels. Although NI Switch Executive checks the route specification string for both valid channel names and syntax, it does not verify that the intermediate channels are reserved for routing, nor does it check for violations of exclusion conflicts.
 - You can programmatically add routes to the route group. However, NI Switch Executive does not check whether the routes can coexist in the route group. NI Switch Executive does not check for the exclusion conflicts or repetitive use of reserved for routing channels.

For best results, open your configuration in MAX, load all routes and route groups in the **Routes/Groups** tab, and run the [validator](#) and/or the [test panel](#) to verify your configuration.

Configuration VIs

Use the VIs on the [Configuration subpalette](#) to build the block diagram.

NI Switch Executive Configuration API

VirtualDevices is a collection of all of the NI Switch Executive configuration objects. An NI Switch Executive **VirtualDevice** object contains the configuration information about a switching system, including the definitions and configurations for the switches, channels, exclusions, hardwires, buses, routes, and route groups associated with the switching configuration:

- **IviDevices** and **IviDevices2** are collections of **IviDevice** and **IviDevice2** objects, respectively. An Ivi switch is a switch module that has an instrument driver that is compliant with the Ivi-8 IviSwch Class Specification as defined by the Ivi Foundation. Ivi-compliant switches include all National Instruments switches as well as some third-party switches.
- **Channels** is a collection of **Channel** objects. A channel is a connection point on a switch. A constituent part of any route, a channel can be associated with a hardwire and/or be a member of an exclusion.
 - **SignalCharacteristics** describe the physical characteristics of NI Switch Executive channels.
- **Exclusions** is a collection of **Exclusion** objects. An exclusion is a user-defined connection rule. The NI Switch Executive routing engine prevents the connection of excluded channels.
- **Hardwires** is a collection of **Hardwire** objects. A hardwire is a user-defined, physical connection between switch channels, such as wiring, analog buses, or matrix expansion plugs. A hardwire can have any number of channels attached to it.
- **Buses** is a collection of **Bus** objects. A bus is a user-defined group of hardwires associated with a single alias name. You can use a bus to quickly create and group similar hardwires.
 - **ChannelsOnBus** is a collection of **BusChannel Pair** objects that defines a bus.
- **Routes** is a collection of **Route** objects. A route is a user-defined connection from one channel (row, column, COM, NO, and so on) to another.
 - **SignalCharacteristics** describe the physical requirements on the constituent channels of a route.
- **RouteGroups** is a collection of **RouteGroup** objects. A route group is a user-defined group of routes and/or other route groups associated with a single alias name. You can use route groups to set up simultaneous switching configurations quickly. Route groups also enable NI Switch Executive to make better decisions about

which resources to use when creating a route.

Data Types for NI Switch Executive Configuration API

Visual Basic	C/C++	Description
Boolean	VARIANT_BOOL	Has the value True (-1) or False (0).
Double	DOUBLE	64-bit floating point number.
Long	LONG	32-bit signed integer.
String	BSTR	A string.
Variant	VARIANT	A variant.

niseCfg_Import

Imports a virtual device configuration from a file.

Function Prototype

NISEStatus __stdcall niseCfg_Import(<i>NISEConstString virtualDeviceName,</i>
	<i>NISEConstString path,</i>
	<i>NISEInt32 overwriteExisting);</i>

Parameter

Input

Name	Type	Description
virtualDeviceName	NISEConstString	The name you want to use for the virtual device configuration.
path	NISEConstString	The full path to the NI Switch Executive virtual device configuration you want to import.
overwriteExisting	NISEInt32	Pass 1 to replace any existing virtual device configurations with the same name. Pass 0 to return an error if a virtual device configuration exists with the same name.

niseCfg_Export

Exports the virtual device configuration to a file.

Function Prototype

NISEStatus __stdcall niseCfg_Export(<i>NISEConstString virtualDeviceName,</i>
	<i>NISEConstString path);</i>

Parameter

Input

Name	Type	Description
virtualDeviceName	NISEConstString	The name you want to use for the virtual device configuration.
path	NISEConstString	The full path to the filename to which you want to export the NI Switch Executive virtual device configuration.

niseCfg_ImportSpecific

Imports a virtual device configuration from a file.

Function Prototype

NISEStatus __stdcall niseCfg_ImportSpecific(<i>NISEConstString virtualDeviceName,</i>
	<i>NISEConstString path,</i>
	<i>NISEInt32 fileFormat,</i>
	<i>NISEInt32 overwriteExisting);</i>

Parameter

Input

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

virtualDeviceName	NISEConstString	The name you want to use for the virtual device configuration.
path	NISEConstString	The full path to the NI Switch Executive virtual device configuration you want to import.
fileFormat	NISEInt32	The format of the file that contains the NI Switch Executive virtual device configuration you want to import.
overwriteExisting	NISEInt32	Pass 1 to replace any existing virtual device configurations with the same name. Pass 0 to return an error if a virtual device configuration exists with the same name.

niseCfg_ExportPrevious

Exports the virtual device configuration to a file format coinciding with a previous version of NI Switch Executive.

Function Prototype

NISEStatus __stdcall niseCfg_ExportPrevious(<i>NISEConstString virtualDeviceName,</i>
	<i>NISEConstString path,</i>
	<i>NISEInt32 previousFileFormat);</i>

Parameter

Input

Name	Type	Description
virtualDeviceName	NISEConstString	The name of the virtual device to export.
path	NISEConstString	The full path to the filename to which you want to export the NI Switch Executive virtual device configuration.
previousFileFormat	NISEInt32	The previous NI Switch Executive version file format to which to export the NI Switch Executive virtual device configuration. The available file formats are as follows: <ul style="list-style-type: none"> NI Switch Executive 2.0 Export File (1)— Exports the NI Switch Executive configuration to an NI Switch

		<p>Executive 2.0 file (XML).</p> <ul style="list-style-type: none"> • NI Switch Executive 2.1 Export File (2)— Exports the NI Switch Executive configuration to an NI Switch Executive 2.1 file (XML). • NI Switch Executive 3.0 Export File (3)— Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 file (XML). • NI Switch Executive 3.0 Excel 97-2003 Workbook (4)— Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 Excel 97-2003 file. • NI Switch Executive 3.0 Excel 2007 Workbook (5)— Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 Excel 2007 file. • NI Switch Executive 3.0 Tab-delimited File (6)— Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 tab delimited file.
--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

niseCfg_ExportSpecific

Exports the virtual device configuration to a file.

Function Prototype

NISEStatus __stdcall niseCfg_ExportSpecific(<i>NISEConstString virtualDeviceName,</i>
	<i>NISEConstString path,</i>
	<i>NISEInt32 fileFormat);</i>

Parameter

Input

Name	Type	Description
virtualDeviceName	NISEConstString	The name you want to use for the virtual device configuration.
path	NISEConstString	The full path to the filename to which you want to export the NI Switch Executive virtual device configuration.

fileFormat	NISEInt32	The format of the file to which you want to export the NI Switch Executive virtual device configuration.
-------------------	-----------	----------------------------------------------------------------------------------------------------------

Bus

A bus is a user-defined group of hardwires associated with a single alias name. You can use a bus to quickly create and group similar hardwires.

Properties

ChannelsOnBus
Comment
Hardwires
HardwireStartIndex
Name
VirtualDevice

ChannelsOnBus Property (Read Only)

Syntax

[Bus.ChannelsOnBus](#)

Data Type

[ChannelsOnBus](#)

Purpose

Returns the collection of BusChannelPair objects for the bus.

Comment Property

Syntax

Bus.Comment

Data Type

String

Purpose

Specifies the comment for the bus.

Hardwires Property (Read Only)

Syntax

Bus.Hardwires

Data Type

Hardwires

Purpose

Returns the collection of the constituent hardwires of the bus.

HardwireStartIndex Property

Syntax

Bus.HardwireStartIndex

Data Type

Long

Purpose

Specifies the start index used to derive hardwire names from the name of the bus.

Name Property

Syntax

Bus.Name

Data Type

String

Purpose

Specifies the bus name.

Remarks

The constituent hardwire names are derived from the bus name by appending a number to the bus name beginning with the HardwireStartIndex value.

VirtualDevice Property (Read Only)

Syntax

Bus.VirtualDevice

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Buses

A collection of [Bus](#) objects.

Properties

Count
Item
VirtualDevice

Methods

Add
Clear
ConvertToHardwires
Remove

See Also

[Bus](#)

Count Property (Read Only)

Syntax

[Buses.Count](#)

Data Type

Long

Purpose

Returns the number of buses in the collection.

Item Property (Read Only)

Syntax

Buses.**Item** (index)

Data Type

Bus

Purpose

Returns the bus object that corresponds to the specified key.

Parameters

index As Variant

[In] Pass the name of the bus or a one-based index into the collection of buses to retrieve the Bus object associated with the specified key. The key name is compared (case-insensitively) to the Bus object names in the collection. The key index is a value between one and the number of buses in a collection. To obtain the number of buses, use the Count property.

VirtualDevice Property (Read Only)

Syntax

Buses.VirtualDevice

Data Type

VirtualDevice

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

Buses.Add (busName)

Return Value

Bus

Purpose

Creates a new Bus object and adds it to the collection.

Remarks

Parameters

busName As String

[In] Pass a unique name for the bus you want to create.

Clear Method

Syntax

Buses.Clear

Purpose

Removes all buses from the collection.

ConvertToHardwires Method

Syntax

Buses.ConvertToHardwires (bus)

Purpose

Converts the bus to hardwires.

Remarks

This method converts all hardwires on the bus to individual, standalone hardwires and deletes the Bus object itself.

Parameters

bus As Bus

[In] Pass the name of the bus or a one-based index into the collection of buses to retrieve the Bus object associated with the specified key. The key name is compared (case-insensitively) to the Bus object names in the collection. The key index is a value between one and the number of buses in a collection. To obtain the number of buses, use the Count property.

Remove Method

Syntax

[Buses.Remove](#) (bus)

Purpose

Removes a bus and all its associated hardwires from the collection.

Remarks

If you want to remove the Bus object without deleting its associated hardwires, use the [ConvertToHardwires](#) method instead.

Parameters

bus As [Bus](#)

[In] Pass the Bus object for the bus you want to remove from the virtual device configuration. To obtain the Bus object, use the [Item](#) property.

BusChannelPair

A pair of [Channel](#) objects that defines a constituent member of a bus.

Properties

[EndChannel](#)

[StartChannel](#)

Methods

[SetChannelRange](#)

StartChannel Property (Read Only)

Syntax

BusChannelPair.StartChannel

Data Type

Channel

Purpose

Returns the first channel of the channel range.

EndChannel Property (Read Only)

Syntax

BusChannelPair.EndChannel

Data Type

Channel

Purpose

Returns the last channel of the channel range.

SetChannelRange Method

Syntax

BusChannelPair.SetChannelRange (startChannel, endChannel)

Purpose

Specifies the first and last channel for the bus channel pair.

Remarks

The startChannel and the endChannel **must** reside on the same IVI device. The IVI physical name of the startChannel **must** alphabetically precede the IVI physical name of the endChannel.

Parameters

startChannel As [Variant](#)

[In] Specifies the first channel of the channel range.

endChannel As [Variant](#)

[In] Specifies the end of the channel range.

Channel

A channel is a connection point on a switch. A constituent part of any route, a channel can be associated with a hardwire and/or be a member of an exclusion.

Properties

Comment
Enabled
FormattedName
Hardwire
IviDevice
Name
ReservedForRouting
SignalCharacteristics
VirtualDevice

Comment Property

Syntax

Channel.**Comment**

Data Type

String

Purpose

Specifies the comment for the channel.

Enabled Property

Syntax

Channel.**Enabled**

Data Type

Boolean

Purpose

Specifies the channel state.

Remarks

If you set this property to `FALSE`, the channel is not considered for any routing task.

FormattedName Property (Read Only)

Syntax

Channel.FormattedName (formatStyle)

Data Type

String

Purpose

Returns the channel name in the specified format.

Parameters

formatStyle As NiseChannelNameStyle

[In] Pass the format style you want to use for the returned channel name string.

Use the following constants with this data type:

- **kAliasAndFull**–Alias and full name
- **kAliasOrFull**–Alias or full name
- **kFullName**–Full name only
- **kIviName**–IVI name only

Hardwire Property (Read Only)

Syntax

Channel.Hardwire

Data Type

Hardwire

Purpose

Returns the hardware associated with the channel.

Remarks

If the channel is not on a hardware, this property returns `NULL`.

IviDevice Property (Read Only)

Syntax

Channel.IviDevice

Data Type

IviDevice

Purpose

Returns the parent Ivi device of the channel.

Name Property

Syntax

Channel.Name

Data Type

String

Purpose

Specifies the alias name for the channel.

Remarks

If you have not specified an alias name for the channel, the alias is an empty string.

ReservedForRouting Property

Syntax

Channel.ReservedForRouting

Data Type

Boolean

Purpose

Specifies the channel usage.

Remarks

If you set this property to `TRUE`, the channel **cannot** be used as an endpoint in any of the routes. However, if you set the channel `ReservedForRouting`, you can use the channel as an intermediary part of the route.

SignalCharacteristics Property (Read Only)

Syntax

Channel.SignalCharacteristics

Data Type

SignalCharacteristics

Purpose

Returns the signal characteristics interface of the channel.

VirtualDevice Property (Read Only)

Syntax

[Channel.VirtualDevice](#)

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Channels

A collection of [Channel](#) objects.

Properties

Count
Item
VirtualDevice

Methods

Add
Clear
Remove

See Also

[Channel](#)

Count Property (Read Only)

Syntax

[Channels.Count](#)

Data Type

[Long](#)

Purpose

Returns the number of channels in the collection.

Item Property (Read Only)

Syntax

[Channels.Item](#) (index)

Data Type

[Channel](#)

Purpose

Returns the Channel object that corresponds to the specified key.

Parameters

index As [Variant](#)

[In] Pass the channel alias, the NI Switch Executive name, or a one-based index to the collection of channels to retrieve the Channel object associated with the specified key. The channel name is compared to the names in the collection. The key index is a value between one and the number of channels in the collection. To obtain the number of channels in the collection, use the [Count](#) property.

VirtualDevice Property (Read Only)

Syntax

[Channels.VirtualDevice](#)

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

[Channels.Add](#) (channel)

Purpose

Adds an existing channel to the collection.

Remarks

Channels **cannot** be added to the virtual device configuration independently. You **must** add the entire IVI device to the configuration to add its channels to the configuration. You can add channels to the collection of hardwires, or to a mutual or

set exclusion.

Parameters

channel As [Channel](#)

[In] Pass the Channel object for the channel you want to add to the collection. To obtain the Channel object, use the [Item](#) property.

Clear Method

Syntax

[Channels.Clear](#)

Purpose

Removes all channels from the collection.

Remarks

You **cannot** use this method to remove a collection of channels on the virtual device. To permanently remove a channel from the virtual device, you **must** remove its parent IVI device from the configuration.

Remove Method

Syntax

[Channels.Remove](#) (index)

Purpose

Removes a channel from the collection.

Parameters

index As [Variant](#)

[In] Pass the channel alias, the NI Switch Executive name, or a one-based index to the collection of channels to retrieve the Channel object associated with the specified key. The channel name is compared (case-insensitively) to the names in the collection. The key index is a value between one and the number of channels in the collection. To obtain the number of channels in the collection, use the [Count](#) property.

ChannelsOnBus

A collection of [BusChannel Pair](#) objects that defines a bus.

Properties

Count

Item

Methods

Add

Clear

Remove

Count Property (Read Only)

Syntax

[ChannelsOnBus.Count](#)

Data Type

[Long](#)

Purpose

Returns the number of channel pairs for the bus in the collection.

Item Property (Read Only)

Syntax

ChannelsOnBus.**Item** (index)

Data Type

BusChannelPair

Purpose

Returns the BusChannelPair object that corresponds to the specified key.

Parameters

index As Variant

[In] Pass the one-based index into the collection of bus channel pairs to retrieve the BusChannelPair object at the specified location in the collection. The key index is a value between one and the number of BusChannelPair objects in the collection. To obtain the number of bus channel pairs in the collection, use the Count property.

Add Method

Syntax

ChannelsOnBus.**Add** (startChannel, endChannel)

Return Value

[BusChannelPair](#)

Purpose

Creates a new BusChannelPair object and adds it to the collection.

Remarks

The startChannel and the endChannel must reside on the same IVI device. The IVI physical name of the startChannel must alphabetically precede the IVI physical name of the endChannel.

Parameters

startChannel As [Channel](#)

[In] Specifies the first channel of the channel range.

endChannel As [Channel](#)

[In] Specifies the last channel of the channel range.

Clear Method

Syntax

[ChannelsOnBus](#).Clear

Purpose

Removes all channel pairs from the bus.

Remove Method

Syntax

ChannelsOnBus.Remove (index)

Purpose

Removes a channel pair from the bus.

Parameters

index As Variant

[In] Pass the one-based index into the collection of bus channel pairs to retrieve the BusChannelPair object at the specified location in the collection. The key index is a value between one and the number of BusChannelPair objects in the collection. To obtain the number of bus channel pairs in the collection, use the Count property.

Exclusion

An exclusion is a user-defined connection rule. The NI Switch Executive routing engine prevents the connection of excluded channels.

Properties

<u>Comment</u>
<u>Enabled</u>
<u>Mutual</u>
<u>Name</u>
<u>Set1</u>
<u>Set2</u>
<u>Type</u>
<u>VirtualDevice</u>

Comment Property

Syntax

Exclusion.**Comment**

Data Type

String

Purpose

Specifies the comment for the exclusion.

Enabled Property

Syntax

Exclusion.**Enabled**

Data Type

Boolean

Purpose

Specifies the exclusion state.

Mutual Property (Read Only)

Syntax

Exclusion.**Mutual**

Data Type

[Channels](#)

Purpose

Returns the interface to the collection of mutually excluded channels.

Remarks

This property only applies when [Type](#) is set to `kMutualExclusion`.

Name Property

Syntax

[Exclusion.Name](#)

Data Type

[String](#)

Purpose

Specifies the exclusion name.

Set1 Property (Read Only)

Syntax

[Exclusion.Set1](#)

Data Type

[Channels](#)

Purpose

Returns the interface to the Set1 collection of excluded channels.

Remarks

This property only applies when Type is set to `kSetExclusion`.

Set2 Property (Read Only)

Syntax

Exclusion.Set2

Data Type

Channels

Purpose

Returns the interface to the Set2 collection of excluded channels.

Remarks

This property only applies when Type is set to `kSetExclusion`.

Type Property (Read Only)

Syntax

Exclusion.Type

Data Type

NiseExclusionType

Returns one of the following constants from this data type:

- **kMutualExclusion**—a list of channels that should never connect to one another.
- **kSetExclusion**—two lists of channels. Channels from the first list should never connect to channels of the second list.

Purpose

Returns the exclusion type.

VirtualDevice Property (Read Only)

Syntax

Exclusion.VirtualDevice

Data Type

VirtualDevice

Purpose

Returns the parent virtual device configuration object.

Exclusions

A collection of Exclusion objects.

Properties

<u>Count</u>
<u>Item</u>
<u>VirtualDevice</u>

Methods

Add
Clear
Remove

See Also

[Exclusion](#)

Item Property (Read Only)

Syntax

[Exclusions](#).**Item** (index)

Data Type

[Exclusion](#)

Purpose

Returns the exclusion object that corresponds to the specified key.

Parameters

index As [Variant](#)

[In] Pass the name of the exclusion or a one-based index into the collection of exclusions to retrieve the Exclusion object associated with the specified key. The key name is compared (case-insensitively) to the Exclusion object names in the collection. The key index is a value between one and the number of exclusions in a collection. To obtain the number of exclusions, use the [Count](#) property.

Count Property (Read Only)

Syntax

Exclusions.Count

Data Type

Long

Purpose

Returns the number of exclusions in the collection.

VirtualDevice Property (Read Only)

Syntax

Exclusions.VirtualDevice

Data Type

VirtualDevice

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

Exclusions.Add (exclusionName, exclusionType)

Return Value

Exclusion

Purpose

Creates a new Exclusion object and adds it to the collection.

Remarks

Parameters

exclusionName As String

[In] Pass a unique name for the exclusion you want to create.

exclusionType As NiseExclusionType

[In] Pass the type of the exclusion you want to create.



Note After the exclusion type is set, you cannot edit it.

Use the following constants with this data type:

- **kMutualExclusion**—list of channels that should never connect to one another.
- **kSetExclusion**—two lists of channels. Channels from the first list should never connect to channels from the second list.

Clear Method

Syntax

Exclusions.Clear

Purpose

Removes all exclusions from the collection.

Remove Method

Syntax

[Exclusions.Remove](#) (index)

Purpose

Removes an exclusion from the collection.

Parameters

index As [Variant](#)

[In] Pass the name of the exclusion or a one-based index into the collection of exclusions to retrieve the Exclusion object associated with the specified key. The key name is compared (case-insensitively) to the Exclusion object names in the collection. The key index is a value between one and the number of exclusions in a collection. To obtain the number of exclusions, use the [Count](#) property.

Hardwire

A hardwire is a user-defined, physical connection between switch channels, such as wiring, analog buses, or matrix expansion plugs. A hardwire can have any number of channels attached to it.

Properties

Channels
Comment
Name
ParentBus
VirtualDevice

Channels Property (Read Only)

Syntax

Hardware.Channels

Data Type

Channels

Purpose

Returns the interface to the collection of channels on the hardware.

Remarks



Note To change the hardware on a channel, first disassociate the channel from its current hardware, and then associate it with another hardware.

Comment Property

Syntax

Hardware.Comment

Data Type

String

Purpose

Specifies the comment for the hardware.

Name Property

Syntax

Hardwire.Name

Data Type

String

Purpose

Specifies the hardware name.

ParentBus Property (Read Only)

Syntax

Hardwire.ParentBus

Data Type

Bus

Purpose

Returns the parent bus object for the hardware.

VirtualDevice Property (Read Only)

Syntax

Hardwire.VirtualDevice

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Hardwires

A collection of [Hardwire](#) objects.

Properties

Count
Item
VirtualDevice

Methods

Add
Clear
Remove

See Also

[Hardwire](#)

Count Property (Read Only)

Syntax

[Hardwires.Count](#)

Data Type

[Long](#)

Purpose

Returns the number of hardwires in the collection.

Item Property (Read Only)

Syntax

[Hardwires](#).**Item** (index)

Data Type

[Hardwire](#)

Purpose

Returns the Hardwire object that corresponds to the specified key.

Parameters

index As [Variant](#)

[In] Pass the name of the hardwire or a one-based index into the collection of hardwires to retrieve the Hardwire object associated with the specified key. The key name is compared (case-insensitively) to the Exclusion object names in the collection. The key index is a value between one and the number of hardwires in a collection. To obtain the number of hardwires in the collection, use the [Count](#) property.

VirtualDevice Property (Read Only)

Syntax

[Hardwires.VirtualDevice](#)

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

[Hardwires.Add](#) (hardwareName)

Return Value

[Hardwire](#)

Purpose

Creates a new Hardwire object and adds it to the collection.

Remarks

Parameters

hardwareName As [String](#)

[In] Pass the name of the hardware you want to add. This name must be unique.

Additionally, hardwire and channel names cannot be the same because hardwires and channels can be used interchangeably.

Clear Method

Syntax

Hardwires.**Clear**

Purpose

Removes all hardwires from the collection.

Remove Method

Syntax

Hardwires.**Remove** (index)

Purpose

Removes a hardwire from the collection.

Parameters

index As Variant

[In] Pass the name of the hardwire or a one-based index into the collection of hardwires to retrieve the Hardwire object associated with the specified key. The key name is compared (case-insensitively) to the Exclusion object names in the collection. The key index is a value between one and the number of hardwires in a collection. To obtain the number of hardwires in the collection, use the Count property.

IviDevice

An IVI switch is a switch module that has an instrument driver that is compliant with

the IVI-8 IviSwch Class Specification as defined by the IVI Foundation. IVI compliant switches include all National Instruments switches as well as some third-party switches.

Properties

Channels
Comment
Name
VirtualDevice

Channels Property (Read Only)

Syntax

[IviDevice.Channels](#)

Data Type

[Channels](#)

Purpose

Returns the collection of channels on the IVI device.

Comment Property

Syntax

[IviDevice.Comment](#)

Data Type

[String](#)

Purpose

Specifies the comment for the IviDevice object.

Name Property (Read Only)

Syntax

IviDevice.Name

Data Type

String

Purpose

Specifies the IVI logical name of the IviDevice object.

VirtualDevice Property (Read Only)

Syntax

IviDevice.VirtualDevice

Data Type

VirtualDevice

Purpose

Returns the parent virtual device configuration object.

IviDevice2

An IVI switch is a switch module that has an instrument driver that is compliant with

the IVI-8 IviSwTch Class Specification as defined by the IVI Foundation. IVI compliant switches include all National Instruments switches as well as some third-party switches.

Properties

Channels
Comment
DriverName
Name
TopologyName
VirtualDevice

Channels Property (Read Only)

Syntax

[IviDevice2.Channels](#)

Data Type

[Channels](#)

Purpose

Returns the collection of channels on the IVI device.

Comment Property

Syntax

[IviDevice2.Comment](#)

Data Type

String

Purpose

Specifies the comment for the IviDevice2 object.

DriverName Property

Syntax

IviDevice2.DriverName

Data Type

String

Purpose

Specifies the driver name for the IVI device. Valid values are "NI-DAQmx" or "IVI".

Name Property (Read Only)

Syntax

IviDevice2.Name

Data Type

String

Purpose

Specifies the IVI logical name of the IviDevice object.

TopologyName Property

Syntax

IviDevice2.TopologyName

Data Type

String

Purpose

Specifies the topology name for the IVI device.

VirtualDevice Property (Read Only)

Syntax

IviDevice2.VirtualDevice

Data Type

VirtualDevice

Purpose

Returns the parent virtual device configuration object.

IviDevices

A collection of IviDevice objects.

Properties

<u>Count</u>
<u>Item</u>

[VirtualDevice](#)

Methods

[Add](#)

[Clear](#)

[Remove](#)

See Also

[IviDevice](#)

Count Property (Read Only)

Syntax

[IviDevices.Count](#)

Data Type

[Long](#)

Purpose

Returns the number of IVI devices in the collection.

Item Property (Read Only)

Syntax

[IviDevices.Item](#) (index)

Data Type

[IviDevice](#)

Purpose

Returns the IviDevice object that corresponds to the specified key.

Parameters

index As [Variant](#)

[In] Pass the name of the IVI device or a one-based index into the collection of IVI devices to retrieve the IviDevice object associated with the specified key. The key name is compared (case-sensitive) to the IviDevice object names in the collection. The key index is a value between one and the number of IVI devices in a collection. To obtain the number of IVI devices in the collection, use the [Count](#) property.

VirtualDevice Property (Read Only)

Syntax

[IviDevices.VirtualDevice](#)

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

`IviDevices.Add (iviLogicalName)`

Return Value

`IviDevice`

Purpose

Adds an IVI device to the virtual device configuration.

Parameters

iviLogicalName As `String`

[In] Pass the IVI Logical Name you defined for the switch device you want to use in the configuration.

Clear Method

Syntax

`IviDevices.Clear`

Purpose

Removes all IVI devices from the virtual device configuration.

Remove Method

Syntax

[IviDevices](#).**Remove** (index)

Purpose

Removes an IVI device from the virtual device configuration.

Parameters

index As [Variant](#)

[In] Pass the name of the IVI device or a one-based index into the collection of IVI devices to retrieve the [IviDevice](#) object associated with the specified key. The key name is compared (case-insensitively) to the [IviDevice](#) object names in the collection. The key index is a value between one and the number of IVI devices in a collection. To obtain the number of IVI devices in the collection, use the [Count](#) property.

IviDevices2

A collection of [IviDevice2](#) objects.

Properties

Count

Item

VirtualDevice

Methods

Add

Clear

Remove

AddEx

See Also

[IviDevice2](#)

Count Property (Read Only)

Syntax

[IviDevices2.Count](#)

Data Type

[Long](#)

Purpose

Returns the number of IVI devices in the collection.

Item Property (Read Only)

Syntax

[IviDevices2.Item](#) (index)

Data Type

[IviDevice2](#)

Purpose

Returns the IviDevice2 object that corresponds to the specified key.

Parameters

index As [Variant](#)

[In] Pass the name of the IVI device or a one-based index into the collection of IVI devices to retrieve the IviDevice2 object associated with the specified key. The key name is compared (case-sensitive) to the IviDevice2 object names in the collection. The key index is a value between one and the number of IVI devices in a collection. To obtain the number of IVI devices in the collection, use the [Count](#) property.

VirtualDevice Property (Read Only)

Syntax

[IviDevices2](#).**VirtualDevice**

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

[IviDevices2](#).**Add** (iviLogicalName)

Return Value

[IviDevice2](#)

Purpose

Adds an IVI device to the virtual device configuration.

Parameters

iviLogicalName As String

[In] Pass the IVI Logical Name you defined for the switch device you want to use in the configuration.

Clear Method

Syntax

IviDevices2.**Clear**

Purpose

Removes all IVI devices from the virtual device configuration.

Remove Method

Syntax

IviDevices2.**Remove** (index)

Purpose

Removes an IVI device from the virtual device configuration.

Parameters

index As Variant

[In] Pass the name of the IVI device or a one-based index into the collection of IVI devices to retrieve the `IviDevice2` object associated with the specified key. The key name is compared (case-insensitively) to the `IviDevice2` object names in the collection. The key index is a value between one and the number of IVI devices in a collection. To obtain the number of IVI devices in the collection, use the [Count](#) property.

AddEx Method

Syntax

[IviDevices2](#).**AddEx** (`iviLogicalName`, `driverName`, `topologyName`)

Return Value

[IviDevice2](#)

Purpose

Adds an IVI device to the virtual device configuration.

Parameters

iviLogicalName As [String](#)

[In] Pass the IVI Logical Name you defined for the switch device you want to use in the configuration.

driverName As [String](#)

[In] Specifies the driver name for the IVI device. Valid values are "NI-DAQmx" or "IVI".

topologyName As [String](#)

[In] Specifies the topology name for the IVI device.

Route

A route is a user-defined connection from one channel (row, column, COM, NO, and so on) to another.

Properties

ChannelsInUse
Comment
Constraints
Endpoint1
Endpoint2
MulticonnectMode
Name
ParentRouteGroups
Specification
SpecificationType
VirtualDevice

ChannelsInUse Property (Read Only)

Syntax

[Route.ChannelsInUse](#)

Data Type

[Channels](#)

Purpose

Returns the channels on the route.

Comment Property

Syntax

Route.Comment

Data Type

String

Purpose

Specifies the comment for the route.

Constraints Property (Read Only)

Syntax

Route.Constraints

Data Type

SignalCharacteristics

Purpose

Returns an interface to the route constraints object.

Endpoint1 Property (Read Only)

Syntax

Route.Endpoint1

Data Type

[Channel](#)

Purpose

Returns the first endpoint of a route.

Endpoint2 Property (Read Only)

Syntax

[Route.Endpoint2](#)

Data Type

[Channel](#)

Purpose

Returns the second endpoint of a route.

MulticonnectMode Property

Syntax

[Route.MulticonnectMode](#)

Data Type

NiseMulticonnectMode

Use the following constants with this data type:

- **kMulticonnectLegs**—Not currently supported by NI Switch Executive.

- **kMulticonnectRoutes**—Routes specified in the connection specification can be connected multiple times. The first call to "Connect" performs the physical hardware connection. Successive calls to "Connect" increase a connection reference count. Similarly, calls to "Disconnect" decrease the reference count. When it reaches 0, the hardware is physically disconnected. Multiconnecting routes applies to entire routes and not to route segments.
- **kNoMulticonnect**—Routes specified in the connection specification must be disconnected before they can be reconnected. Calling "Connect" on a route that was connected using No Multiconnect mode results in an error.

Purpose

Specifies the multiconnect mode for the route.

Name Property

Syntax

Route.Name

Data Type

String

Purpose

Specifies the route name.

ParentRouteGroups Property (Read Only)

Syntax

Route.ParentRouteGroups

Data Type

RouteGroups

Purpose

Returns the collection of parent route groups for the route.

Specification Property

Syntax

Route.Specification

Data Type

String

Purpose

Specifies the route specification.

Remarks

Note The following actions do not cause an error in the configuration API:

- Route creation that uses channels with different wire modes
- Use of valid syntax, without reserved for routing checks of the intermediate channels or checks of exclusion violation conflicts



To verify the configuration, open the configuration in MAX, load all routes and route group in the **Routes/Groups** tab, and [validate](#) and/or run the [test panel](#).

SpecificationType Property

Syntax

Route.SpecificationType

Data Type

NiseRouteSpecificationType

Use the following constants with this data type:

- **kSpecifiedByBoth**—If the full path fails to connect, the routing engine uses endpoints to determine the route.
- **kSpecifiedByEndpoints**—The routing engine uses endpoints to determine the route.
- **kSpecifiedByFullPath**—The routing engine uses the fully specified path to determine the route.

Purpose

Specifies the connect method the routing engine uses to determine a route.

VirtualDevice Property (Read Only)

Syntax

Route.VirtualDevice

Data Type

VirtualDevice

Purpose

Returns the parent virtual device configuration object.

Routes

A collection of [Route](#) objects.

Properties

Count
Item
VirtualDevice

Methods

Add
Clear
Remove

See Also

[Route](#)

Count Property (Read Only)

Syntax

[Routes.Count](#)

Data Type

[Long](#)

Purpose

Returns the number of routes in the collection.

Item Property (Read Only)

Syntax

Routes.Item (index)

Data Type

Route

Purpose

Returns the route object that corresponds to the specified key.

Parameters

index As Variant

[In] Pass the name of the route or a one-based index into the collection of routes to retrieve the Route object associated with the specified key. The key name is compared (case-insensitively) to the Route object names in the collection. The key index is a value between one and the number of routes in a collection. To obtain the number of routes in the collection, use the Count property.

VirtualDevice Property (Read Only)

Syntax

Routes.VirtualDevice

Data Type

VirtualDevice

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

Routes.Add (route)

Return Value

Route

Purpose

Creates a new Route object and adds it to the collection.

Remarks

If you pass an existing Route object for the route parameter, the existing route is added to the collection.

Parameters

route As Variant

[In] Pass a unique name for the route you want to create. Routes and RouteGroups **cannot** have the same name. Pass an existing route to add the route to the collection.



Note You can only add new routes to the collection of routes on a virtual device. Conversely, you can only add an existing route to the collection of routes on a route group.

Clear Method

Syntax

[Routes.Clear](#)

Purpose

Removes all routes from the collection.

Remove Method

Syntax

[Routes.Remove](#) (index)

Purpose

Removes a route from the collection.

Parameters

index As [Variant](#)

[In] Pass the name of the route or a one-based index into the collection of routes to retrieve the Route object associated with the specified key. The key name is compared (case-insensitively) to the Route object names in the collection. The key index is a value between one and the number of routes in a collection. To obtain the number of routes in the collection, use the [Count](#) property.

RouteGroup

A route group is a user-defined group of routes and/or other route groups associated with a single alias name. You can use route groups to set up simultaneous switching configurations quickly. Route groups also enable NI Switch Executive to make better decisions about which resources to use when creating a route.

Properties

ChildRouteGroups
ChildRoutes
Comment
Name
VirtualDevice

ChildRoutes Property (Read Only)

Syntax

[RouteGroup](#).**ChildRoutes**

Data Type

[Routes](#)

Purpose

Returns the collection of child routes for the route.

ChildRouteGroups Property (Read Only)

Syntax

[RouteGroup](#).**ChildRouteGroups**

Data Type

[RouteGroups](#)

Purpose

Returns the collection of child route groups for the route.

Comment Property

Syntax

RouteGroup.Comment

Data Type

String

Purpose

Specifies the comment for the route group.

Name Property

Syntax

RouteGroup.Name

Data Type

String

Purpose

Specifies the route group name.

VirtualDevice Property (Read Only)

Syntax

RouteGroup.VirtualDevice

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

RouteGroups

A collection of [RouteGroup](#) objects.

Properties

Count
Item
VirtualDevice

Methods

Add
Clear
Remove

See Also

[RouteGroup](#)

Count Property (Read Only)

Syntax

[RouteGroups.Count](#)

Data Type

[Long](#)

Purpose

Returns the number of route groups in the collection.

Item Property (Read Only)

Syntax

[RouteGroups.Item](#) (index)

Data Type

[RouteGroup](#)

Purpose

Returns the RouteGroup object that corresponds to the specified key.

Parameters

index As [Variant](#)

[In] Pass the name of the route group or a one-based index into the collection of route groups to retrieve the RouteGroup object associated with the specified key. The key name is compared (case-insensitively) to the RouteGroup object names in the collection. The key index is a value between one and the number of routes in a collection. To obtain the number of route groups, use the [Count](#) property.

VirtualDevice Property (Read Only)

Syntax

[RouteGroups.VirtualDevice](#)

Data Type

[VirtualDevice](#)

Purpose

Returns the parent virtual device configuration object.

Add Method

Syntax

[RouteGroups.Add](#) (routeGroup)

Return Value

[RouteGroup](#)

Purpose

Creates a new RouteGroup object and adds it to the collection.

Remarks

Pass an existing RouteGroup object to add it to the collection.



Note NI Switch Executive does **not** detect and report reserved for routing conflicts or exclusion conflicts when you attempt to associate a route or a route group with another route group. To verify the configuration, open the configuration in MAX, load all routes and route

group in the **Routes/Groups** tab, and run the validator and/or the test panel.

Parameters

routeGroup As [Variant](#)

[In] Pass a unique name for the route group you want to create. Routes and route groups **cannot** have the same name. Pass an existing route group to add the route group to the collection.



Note You can only add new route groups to the collection of routes on a virtual device. Conversely, you can only add an existing route group to the collection of child route groups on a route group.

Clear Method

Syntax

[RouteGroups](#).**Clear**

Purpose

Removes all route groups from the collection.

Remove Method

Syntax

[RouteGroups](#).**Remove** (index)

Purpose

Removes a route group from the collection.

Parameters

index As [Variant](#)

[In] Pass the name of the route group or a one-based index into the collection of route groups to retrieve the RouteGroup object associated with the specified key. The key name is compared (case-insensitively) to the RouteGroup object names in the collection. The key index is a value between one and the number of routes in a collection. To obtain the number of route groups, use the [Count](#) property.

SignalCharacteristics

Signal characteristics describe the physical characteristics of NI Switch Executive channels. When you use this object in the context of a route, the signal characteristics describe the physical requirements on the constituent channels.

Properties

Bandwidth
FormattedValue
Impedance
MaxAcCarryCurrent
MaxAcCarryPower
MaxAcSwitchingCurrent
MaxAcSwitchingPower
MaxAcVoltage
MaxDcCarryCurrent
MaxDcCarryPower
MaxDcSwitchingCurrent
MaxDcSwitchingPower
MaxDcVoltage
SettlingTime
WireMode

Bandwidth Property

Syntax

SignalCharacteristics.**Bandwidth**

Data Type

Double

Purpose

Returns/specifies the bandwidth characteristic/constraint.

FormattedValue Property (Read Only)

Syntax

SignalCharacteristics.**FormattedValue** (attribute)

Data Type

String

Purpose

Returns the formatted characteristic/constraint that includes the value and the unit.

Parameters

attribute As NiseSignalCharacteristic

[In] Pass the type of the signal characteristics for which you want to retrieve the value.

Use the following constants with this data type:

- **kBandwidth**–Bandwidth
- **kImpedance**–Impedance
- **kMaxAcCarryCurrent**–Max AC Carry Current
- **kMaxAcCarryPower**–Max AC Carry Power
- **kMaxAcSwitchingCurrent**–Max AC Switching Current
- **kMaxAcSwitchingPower**–Max AC Switching Power
- **kMaxAcVoltage**–Max AC Voltage
- **kMaxDcCarryCurrent**–Max DC Carry Current
- **kMaxDcCarryPower**–Max DC Carry Power
- **kMaxDcSwitchingCurrent**–Max DC Switching Current
- **kMaxDcSwitchingPower**–Max DC Switching Power
- **kMaxDcVoltage**–Max DC Voltage
- **kSettlingTime**–SettlingTime
- **kWireMode**–Wire Mode

Impedance Property

Syntax

SignalCharacteristics.**Impedance**

Data Type

Double

Purpose

Returns/specifies the impedance characteristic/constraint.

MaxAcCarryCurrent Property

Syntax

SignalCharacteristics.**MaxAcCarryCurrent**

Data Type

Double

Purpose

Returns/specifies the maximum AC carry current characteristic/constraint.

MaxDcCarryCurrent Property

Syntax

SignalCharacteristics.MaxDcCarryCurrent

Data Type

Double

Purpose

Returns/specifies the maximum DC carry current characteristic/constraint.

MaxAcCarryPower Property

Syntax

SignalCharacteristics.MaxAcCarryPower

Data Type

Double

Purpose

Returns/specifies the maximum AC carry power characteristic/constraint.

MaxDcCarryPower Property

Syntax

SignalCharacteristics.MaxDcCarryPower

Data Type

Double

Purpose

Returns/specifies the maximum DC carry power characteristic/constraint.

MaxAcSwitchingCurrent Property

Syntax

SignalCharacteristics.MaxAcSwitchingCurrent

Data Type

Double

Purpose

Returns/specifies the maximum AC switching current characteristic/constraint.

MaxDcSwitchingCurrent Property

Syntax

SignalCharacteristics.MaxDcSwitchingCurrent

Data Type

Double

Purpose

Returns/specifies the maximum DC switching current characteristic/constraint.

MaxAcSwitchingPower Property

Syntax

SignalCharacteristics.MaxAcSwitchingPower

Data Type

Double

Purpose

Returns/specifies the maximum AC switching power characteristic/constraint.

MaxDcSwitchingPower Property

Syntax

SignalCharacteristics.MaxDcSwitchingPower

Data Type

Double

Purpose

Returns/specifies the maximum DC switching power characteristic/constraint.

MaxAcVoltage Property

Syntax

SignalCharacteristics.MaxAcVoltage

Data Type

Double

Purpose

Returns/specifies the maximum AC voltage characteristic/constraint.

MaxDcVoltage Property

Syntax

SignalCharacteristics.MaxDcVoltage

Data Type

Double

Purpose

Returns/specifies the maximum DC voltage characteristic/constraint.

SettlingTime Property

Syntax

SignalCharacteristics.SettlingTime

Data Type

[Double](#)

Purpose

Returns/specifies the settling time characteristic/constraint.

WireMode Property

Syntax

[SignalCharacteristics.WireMode](#)

Data Type

[Long](#)

Purpose

Returns/specifies the wire mode characteristic/constraint.

VirtualDevice

An NI Switch Executive virtual device contains the configuration information about a switching system, including the definitions and configurations for the switches, channels, hardwires, buses, exclusions, routes, and route groups associated with the switching configuration.

Properties

Buses
Channels
Comment
Exclusions
Hardwires
IviDevices

Name
RouteGroups
Routes
Version

Buses Property (Read Only)

Syntax

[VirtualDevice](#).**Buses**

Data Type

[Buses](#)

Purpose

Returns the collection of buses for the virtual device configuration.

Channels Property (Read Only)

Syntax

[VirtualDevice](#).**Channels**

Data Type

[Channels](#)

Purpose

Returns the collection of channels for the virtual device configuration.

Comment Property

Syntax

VirtualDevice.Comment

Data Type

String

Purpose

Specifies the comment for the virtual device configuration.

Exclusions Property (Read Only)

Syntax

VirtualDevice.Exclusions

Data Type

Exclusions

Purpose

Returns the collection of exclusions for the virtual device configuration.

Hardwires Property (Read Only)

Syntax

VirtualDevice.Hardwires

Data Type

Hardwires

Purpose

Returns the collection of hardwires for the virtual device configuration.

IviDevices Property (Read Only)

Syntax

VirtualDevice.IviDevices

Data Type

IviDevices

Purpose

Returns the collection of IVI device objects for the virtual device configuration.

Name Property

Syntax

VirtualDevice.Name

Data Type

String

Purpose

Specifies the name of the virtual device configuration.

RouteGroups Property (Read Only)

Syntax

VirtualDevice.RouteGroups

Data Type

RouteGroups

Purpose

Returns the collection of route groups for the virtual device configuration.

Routes Property (Read Only)

Syntax

VirtualDevice.Routes

Data Type

Routes

Purpose

Returns the collection of routes for the virtual device configuration.

Version Property

Syntax

VirtualDevice.Version

Data Type

[String](#)

Purpose

Specifies the version of the virtual device configuration.

VirtualDevices

A top-level collection of all of the NI Switch Executive configurations.

Properties

Count
Item

Methods

Add
Remove
Save

See Also

[VirtualDevice](#)

Count Property (Read Only)

Syntax

[VirtualDevices.Count](#)

Data Type

Long

Purpose

Returns the number of virtual device configurations.

Item Property (Read Only)

Syntax

VirtualDevices.**Item** (index)

Data Type

VirtualDevice

Purpose

Returns the virtual device configuration that corresponds to the specified key.

Parameters

index As Variant

[In] Pass the name of the virtual device or a one-based index into the collection of virtual devices to retrieve the VirtualDevice object associated with the specified key. The key name is compared (case-insensitively) to the VirtualDevice object names in the collection. The key index is a value between one and the number of virtual devices in a collection. To obtain the number of route groups in the collection, use the Count property.

Add Method

Syntax

VirtualDevices.Add (virtualDeviceName)

Return Value

VirtualDevice

Purpose

Creates a new, empty virtual device configuration.

Remarks

Parameters

virtualDeviceName As String

[In] Pass the name you want to assign the new virtual device.

Save Method

Syntax

VirtualDevices.Save

Purpose

Saves all the changes to the disk.

Remove Method

Syntax

`VirtualDevices.Remove (index)`

Purpose

Deletes the virtual device.

Parameters

index As [Variant](#)

[In] Pass the name of the virtual device or a one-based index into the collection of virtual devices to retrieve the `VirtualDevice` object associated with the specified key. The key name is compared (case-insensitively) to the `VirtualDevice` object names in the collection. The key index is a value between one and the number of virtual devices in a collection. To obtain the number of route groups in the collection, use the [Count](#) property.

Using the Configuration API in Visual C++

Include the following lines in your code to use the NI Switch Executive Configuration API from Visual C++:

```
#include "comdef.h"
```

```
#import "C:\windows\system32\nise.dll" tlbid(2) no_namespace
```



Note The `tlbid(2)` attribute is important because the COM type library is the second resource in a DLL. The first `typelib` resource is the NI Switch Executive C API type library for VisualBasic. The `no_namespace` attribute designates that you can use the NI Switch Executive classes without specifying a namespace.

Example Usage

```
#include "stdafx.h"

#include "comdef.h"

#import "C:\windows\system32\nise.dll" tlbid(2) no_namespace

int _tmain(int argc, _TCHAR* argv[])

{

HRESULT hr = CoInitialize(NULL);

if (SUCCEEDED(hr))

{

VirtualDevicesPtr nise(__uuidof(NiseVirtualDevices)); // cr
eates a smart pointer, no need to destroy this one

VirtualDevice* vd; // this will hold a pointer to VirtualDe
vice interface.

long numDevices;

BSTR vdName;

numDevices = nise->GetCount(); // get the number of virtual
devices. You should have one (the example one) after defaul
t installation

vd = nise->GetItem(1); // get the first element in the coll
ection of virtual devices. You can loop from 1 to numDevice
s if you want to examine the whole system

vdName = vd->GetName(); //get the name of the first configu
```

ration in a collection

```
SysTreeString(vdName);
```

```
vd->Release(); // vd is not a smart pointer, you should release it. Smart pointers end in "Ptr"
```

```
CoUninitialize();
```

```
}
```

```
return 0;
```

```
}
```

Using the Configuration API in Excel

The use of the configuration API in Microsoft Excel was first available in NI Switch Executive 2.1. For more information about using the configuration API in Excel, visit ni.com/info and enter the Info Code NISEConfig.

Managing a Virtual Device in Excel

NI Switch Executive includes options in MAX and functions in the [Configuration API to export](#) and manage a virtual device in Microsoft Excel. You can use Excel to scale, customize, and streamline NI Switch Executive configuration [management](#) and perform any of the following tasks:

- Mass rename devices or channels
- Mass define routes
- Store non-configuration information (pinout and other system data) with NI Switch Executive configuration information in one file

When you have completed your management tasks in Excel, you can [import](#) the modified configuration to continue working with the virtual device in other environments/interfaces.

Exporting to Excel

To manage configuration data in Excel, first [export](#) the virtual device to an Excel-compatible file format.

The following table lists the objects (worksheets) and attributes (columns) that NI Switch Executive generates when exporting to Excel.



Note NI Switch Executive requires some attributes (columns) to successfully import. Required attributes (columns) are called **key columns**.

Object (Worksheet)	Attribute (Column)
Overview	N/A
Virtual Device	Virtual Device Name, Comment, Export Schema Version, Channel Name Style, Route Editor Channel Display, Route Editor Route Updates, Route Editor Route Search Depth, Schematic Route Display, Schematic Route Deletion
IVI Devices	IVI Device Name, Comment
Channels	Channel IVI Device, Channel IVI Name, Alias, Hardwire, Reserved For Routing, Disabled, Comment, Wire Mode, Settling Time (s), Impedance (Ohm), Bandwidth (Hz), DC Voltage (V), AC Voltage (V), DC Switching Current (A),

	AC Switching Current (A), DC Carry Current (A), AC Carry Current (A), DC Switching Power (W), AC Switching Power (W), DC Carry Power (W), AC Carry Power (W)
Routes	Route Name, Endpoint1, Endpoint2, Specification, Type, Multiconnect Mode, Comment, Wire Mode, Settling Time (s), Impedance (Ohm), Bandwidth (Hz), DC Voltage (V), AC Voltage (V), DC Switching Current (A), AC Switching Current (A), DC Carry Current (A), AC Carry Current (A), DC Switching Power (W), AC Switching Power (W), DC Carry Power (W), AC Carry Power (W)
Route Groups	Route Group Name, Comment
Route Group Children	Route Group Name, Child
Hardwires	Hardwire Name, Comment
Exclusions	Exclusion Name, Type, Disabled, Comment
Mutual Exclusion Channels	Exclusion Name, Channel

Set Exclusion Set1 Channels	Exclusion Name, Set1 Channel
Set Exclusion Set2 Channels	Exclusion Name, Set2 Channel
Buses	Bus Name, Hardwire Name Start Index, Comment
Bus Channels	Bus Name, IVI Device, Base Channel, End Channel

Excel Workbooks

When you export to Excel workbooks (.xls and .xlsx), NI Switch Executive saves the configuration data of a virtual device, organized by object type, in a multi-sheet **workbook**. Each **worksheet** in the workbook contains the configuration data stored in **columns** and **rows**.

The column headings specify the attributes associated with the object. Thereafter, each row in a worksheet represents a separate NI Switch Executive object and each column represents an attribute value associated with that object.



Note NI Switch Executive requires some attributes (columns) to successfully import. Refer to [Formatting Configuration Data](#) for more information about required attributes (columns).

Example

The following figure shows an IVI Devices worksheet. The column headings, IVI Logical Name and Comment, specify the attributes associated with an IVI Device object. The following rows define actual IVI devices, such as PXI1Slot2 and SampleMatrix1, and specify the attribute values associated with those devices.

IVI Logical Name	Comment
PXI1Slot2	PXI-2570
PXI1Slot2_ivi	PXI-2570
SampleMatrix1	Sample Switch (8x8 Matrix)
SampleMatrix2	Sample Switch (8x8 Matrix)

Tab-Delimited Text Files

When you export to a tab-delimited (.txt) file, NI Switch Executive saves the configuration data of a virtual device in a single Excel-compatible file. When you open a tab-delimited file in Excel, NI Switch Executive objects, organized by object type, are separated by an empty row. The first row immediately following the empty row represents the column headings for that object.



Note NI Switch Executive requires some attributes (columns) to successfully import. Refer to [Formatting Configuration Data](#) for more information about required attributes (columns).

Formatting Configuration Data

NI Switch Executive organizes the configuration data associated with a virtual device in the columns and rows of a worksheet. To modify and add data without affecting the ability to import the file, adhere to the following guidelines.



Note If you fail to format data in Excel in accordance with the following rules and guidelines, NI Switch Executive returns an error at import, and import fails.



Note When importing data, ensure that all required attributes (columns) are specified.

Rows

- **Inserting a row**—You can insert a row anywhere in a workbook.
 - NI Switch Executive requires an empty row between configuration data and any custom non-configuration data you add to a worksheet.
 - NI Switch Executive does not require an empty row between a column heading row and its data rows.

Columns

- **Inserting a column**—You can insert a column anywhere in a workbook.
 - NI Switch Executive accepts empty columns.
- **Deleting a column**—You can only delete a column in a workbook if it is not a required column. All other columns are optional.
- **Renaming a column**—Not supported, unless the column was added by a user.
- **Reordering columns**—You can reorder any column in a workbook.

Worksheets

- **Inserting a worksheet**—You can insert a worksheet or a custom sheet, such as a chart sheet, macro, or a dialog sheet, anywhere in a workbook.
- **Inserting a chart or object**—You can insert a chart or other object anywhere in a worksheet.
- **Deleting a worksheet**—You can only delete a worksheet if it is not the IVI Device worksheet. The IVI Device worksheet is required when you import; all other worksheets are optional.
- **Renaming a worksheet**—You can rename any worksheet.
- **Reordering worksheets**—You can reorder any worksheet.

Workbooks

- **Renaming a workbook**—You can rename a workbook.

Formatting Configuration Data: Virtual Device

Use the following guidelines when formatting a Virtual Device in Excel.



Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at [import](#). If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Virtual Device Name	String	No	Value specified in Import dialog or API
Comment	String Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.	No	Empty String
Export Schema Version	1	No	1
Channel Name Style	Full Name Only Alias Or Full Name Alias And Full Name	No	Alias Or Full Name
Route Editor Channel Display	Show Channels Without Aliases Hide Channels Without Aliases	No	Show Channels Without Aliases
Route Editor Route Updates	Auto Update Routes Manual Update Routes	No	Auto Update Routes
Route Editor Route Search Depth	Low Medium High	No	Low
Schematic Route Display	Hide Routes Show Routes	No	Hide Routes
Schematic Route Deletion	Show Notification Dialog Do Not Show Notification Dialog	No	Show Notification Dialog

Formatting Configuration Data: IVI Devices

Use the following guidelines when formatting IVI Devices in Excel.




Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at [import](#). If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the

	attribute at import.
--	----------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
IVI Device Name	String	Yes	None
Comment	String Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.	No	Empty String

Formatting Configuration Data: Channels

Use the following guidelines when formatting Channels in Excel.

	Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at import . If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Channel IVI Device	String	Yes	None
Channel IVI Name	String	Yes	None
Alias	String	No	Empty String
Hardwire	String	No	Empty String
Reserved For Routing	Empty String or x Notice: Empty String means not reserved for routing	No	Empty String
Disabled	Empty String or x Notice: Empty String means not disabled	No	Empty String
Comment	String	No	Empty String

	Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.		
Wire Mode	1, 2, or 4	No	N/A
Settling Time (s)	Float	No	N/A
Impedance (Ohm)	Float	No	N/A
Bandwidth (Hz)	Float	No	N/A
DC Voltage (V)	Float	No	N/A
AC Voltage (V)	Float	No	N/A
DC Switching Current (A)	Float	No	N/A
AC Switching Current (A)	Float	No	N/A
DC Carry Current (A)	Float	No	N/A
AC Carry Current (A)	Float	No	N/A
DC Switching Power (W)	Float	No	N/A
AC Switching Power (W)	Float	No	N/A
DC Carry Power (W)	Float	No	N/A
AC Carry Power (W)	Float	No	N/A

Formatting Configuration Data: Routes

Use the following guidelines when formatting Routes in Excel.



Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at [import](#). If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.


Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Route Name	String	Yes	None
Endpoint1	String	Yes*	Uses route spec
Endpoint2	String	Yes*	Uses route spec
Specification	String	Yes*	Uses endpoints
Type	Specified By Full Path, Find By Endpoints, Find If Full Path Fails	Yes	None
Multiconnect Mode	Multiconnect, No Multiconnect	Yes	None
Comment	String Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.	No	Empty String
Wire Mode	1, 2, or 4	No	Uses Endpoint1 or route specification
Settling Time (s)	Float	No	None
Impedance (Ohm)	Float	No	None
Bandwidth (Hz)	Float	No	None
DC Voltage (V)	Float	No	None
AC Voltage (V)	Float	No	None
DC Switching Current (A)	Float	No	None
AC Switching Current (A)	Float	No	None
DC Carry Current (A)	Float	No	None
AC Carry Current (A)	Float	No	None
DC Switching Power (W)	Float	No	None
AC Switching Power (W)	Float	No	None

DC Carry Power (W)	Float	No	None
AC Carry Power (W)	Float	No	None

* One of these is required: either both endpoints or the specification. When all three are specified, the specification is used before the endpoints.

Formatting Configuration Data: Route Groups


Use the following guidelines when formatting Route Groups in Excel.

	Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at import . If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Route Group Name	String	Yes	None
Comment	String Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.	No	Empty String

Formatting Configuration Data: Route Group Children

Use the following guidelines when formatting Route Group Children in Excel.


	Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at import . If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.
-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
--------------------	--------------	----------	-------------------------------

Route Group Name	String	Yes	None
Child	String	Yes	None

Formatting Configuration Data: Hardwires


Use the following guidelines when formatting Hardwires in Excel.

	Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at import . If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Hardwire Name	String	Yes	None
Comment	String Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.	No	Empty String

Formatting Configuration Data: Exclusions

Use the following guidelines when formatting Exclusions in Excel.


	Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at import . If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.
-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Exclusion Name	String	Yes	None
Type	Mutual, Set	No	Mutual
Disabled	Empty String or x	No	Empty String

	Notice: Empty String means not disabled.		
Comment	String Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.	No	Empty String

Formatting Configuration Data: Mutual Exclusion Channels


Use the following guidelines when formatting Mutual Exclusion Channels in Excel.

	Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at import . If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Exclusion Name	String	Yes	None
Channel	String	Yes	None

Formatting Configuration Data: Set Exclusion Set1 Channels

Use the following guidelines when formatting Set Exclusion Set1 Channels in Excel.

	Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at import . If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.
-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Exclusion Name	String	Yes	None
Set1 Channel	String	Yes	None

Formatting Configuration Data: Set Exclusion Set2 Channels

Use the following guidelines when formatting Set Exclusion Set2 Channels in Excel.



Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at [import](#). If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Exclusion Name	String	Yes	None
Set2 Channel	String	Yes	None

Formatting Configuration Data: Buses

Use the following guidelines when formatting Buses in Excel.



Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at [import](#). If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Bus Name	String	Yes	None
Hardwire Name Start Index	Integer	No	0
Comment	String Notice: Use ^t, ^r, and ^n for the tab, carriage return, and new line characters, respectively.	No	Empty String

Formatting Configuration Data: Bus Channels

Use the following guidelines when formatting Bus Channels in Excel.



Note NI Switch Executive replaces the attribute value with "Imported Value if Unspecified" only at [import](#). If "Imported Value if Unspecified" is N/A, NI Switch Executive ignores the attribute at import.

Attribute (Column)	Valid Values	Required	Imported Value if Unspecified
Bus Name	String	Yes	None
IVI Device	String	Yes	None
Base Channel	String	Yes	None
End Channel	String	Yes	None

Adding Custom Data

You can add your own custom data and charts. This data is ignored at import but is saved in the workbook.

Performing Common Management Tasks

You can modify NI Switch Executive data in Excel to create custom configurations. To modify data in Excel without affecting the ability to import the file, always populate required columns and [format](#) the data appropriately.

Refer to the following management task cases for instructions about how to customize NI Switch Executive data in Excel.



Note For instructions about performing specific tasks in Excel (how to mass populate a column, how to search and replace, and more), refer to the ***Excel Help*** included with Excel.

- [Mass Renaming Channel Aliases](#)
- [Mass Creating Routes](#)
- [Editing an IVI Device Name](#)
- [Changing an IVI Device Topology](#)
- [Merging Two Virtual Devices](#)

Mass Renaming Channel Aliases

Complete the following steps to mass rename channel aliases:

1. Click the **Channels** sheet tab at the bottom of the screen to display the Channels worksheet.

2. Populate the **Alias** column with aliases for each Channel IVI Device/Channel IVI Name you want to alias.
3. Save the edits.

Mass Creating Routes

Complete the following steps to mass create routes:

1. Click the **Routes** sheet tab at the bottom of the screen to display the Routes worksheet.
2. Insert a row for each route you want to create, and populate the required attributes (columns):
 - Route Name
 - Either both Endpoint1 and Endpoint2, or Specification, or all three
 - Type
 - Multiconnect Mode

Refer to [Formatting Configuration Data: Routes](#) for detailed information about formatting Route attributes (columns).

3. (Optional) Populate other attributes (columns).
4. Save the edits.

Editing an IVI Device Name

Globally Editing Device References

Complete the following steps to globally edit device references:

1. Open a search and replace for the entire workbook.
2. Specify the existing IVI device name as the search string and the replacing IVI device name as the replace string. Excel edits every device reference in the workbook.
3. Save the edits.

Individually Editing Device References

Complete the following steps to individually edit device references:

1. Open a search.
2. Specify the existing IVI device name as the search string. Excel finds an instance of the IVI Device Name in the workbook.
3. Edit the device reference.
4. Repeat steps 2 and 3 until you have edited every device reference in the workbook.
5. Save the edits.



Note Device references display in the IVI Device Name, Channel IVI Device, IVI Device, Endpoint1, Endpoint2, and Specification columns of the IVI Devices, Channels, Bus Channels, and Routes worksheets.

Changing an IVI Device Topology

Complete the following steps to change an IVI device topology:

1. Add new channels.
 1. Click the **Channels** sheet tab at the bottom of the screen to display the Channels worksheet.
 2. Insert a row for each channel you want to create and populate the required attributes (columns):
 - Channel IVI Device
 - Channel IVI Name
 Refer to [Formatting Configuration Data: Channels](#) for detailed information about formatting Channel attributes (columns).
 3. (Optional) Populate other attributes (columns).
2. Replace existing channel references with new channel references using one of the following methods:
 - **Globally editing channel references**—Complete the following steps to globally edit channel references:
 1. Open a search and replace for the entire workbook.
 2. Specify the existing IVI Channel name as the search string and the replacing IVI Channel name as the replace string. Excel edits every channel reference in the workbook.
 3. Save the edits.
 - **Individually editing channel references**—Complete the following steps to individually edit channel references:
 1. Open a search.

2. Specify the existing IVI Channel name as the search string. Excel finds an instance of the IVI Channel Name in the workbook.
3. Edit the channel reference.
4. Repeat steps b and c until you have edited every device reference in the workbook.
5. Save the edits.



Note Channel references display in the Channel IVI Name, Channel, Set1 Channel, Set2 Channel, Base Channel, End Channel, Endpoint1, Endpoint2, and Specification columns of the Channels, Mutual Exclusion Channels, Set Exclusion Set1 Channels, Set Exclusion Set2 Channels, Bus Channels, and Routes worksheets.

3. Delete unused channels and channel references.
 1. Click the **Channels** sheet tab at the bottom of the screen to display the Channels worksheet.
 2. Delete the row for any IVI channel that does not exist on the new topology.
 3. Delete any rows in other worksheets that contain unused channel references.
4. Save the edits.



Note Ensure that the topology in the IVI Configuration in MAX is updated to match the new topology.

Merging Two Virtual Devices

Complete the following steps to create one virtual device from two:

1. Merge the virtual devices.
 1. Open the workbooks for both Virtual Device1 and Virtual Device2.
 2. Click a worksheet tab at the bottom of the screen in Virtual Device2.
 3. Copy the rows of entire worksheet (excluding column headers).
 4. Insert the copied rows into the same worksheet in the Virtual Device1 workbook. Virtual Device1 now contains the rows of both Virtual Device1 and Virtual Device2.
 5. Repeat steps b through d for each worksheet (excluding the Virtual Device worksheet) in Virtual Device2.
 6. Close the Virtual Device2 workbook. Virtual Device1 contains the merged virtual device data.
2. Delete any redundant rows (IVI devices) on the IVI Devices worksheet.

3. Resolve any reference conflicts.



Note References display in the Alias, Hardwire, Exclusion Name, Channel, Set1Channel, Set2 Channel, Hardwire Name, Bus Name, Base Channel, End Channel, Route Name, Endpoint1, Endpoint2, Specification, Route Group Name, Child columns of the workbook.

1. Excluding the Virtual Device and IVI Devices worksheets, verify that all references are unique in the following columns of the workbook:
 - Alias
 - Exclusion Name
 - Hardwire Name
 - Bus Name
 - Route Name
 - Route Group Name
2. If necessary, replace non-unique references in the workbook.
4. Save the edits.

Importing Excel Data

After you have customized the virtual device configuration in Excel, [import](#) the configuration to continue working with the virtual device in other environments/interfaces.



Note NI Switch Executive preserves the state of your file at import to preserve custom, non-configuration data.

Troubleshooting

If you experience an error at import, verify that you have customized data in Excel in accordance with the [formatting](#) rules and guidelines.

Programming with NI Switch Executive

NI Switch Executive provides full programmatic control of your virtual device using an application development environment (ADE). Refer to [Getting Started](#) to begin using your virtual device in LabVIEW, NI TestStand, LabWindows/CVI, or Visual Basic.

Getting Started

You can find information about how to begin using NI Switch Executive with your ADE, any files to include in your application, and considerations for each ADE in this topic.

To successfully build your application, you must have NI Switch Executive and one of the following ADEs installed:

- [LabVIEW](#)
- [LabWindows/CVI](#)
- [Visual C++](#)
- [Visual Basic](#)
- [NI TestStand](#)

Using NI Switch Executive in LabVIEW

This topic assumes that you are using LabVIEW to manage your code development and that you are familiar with the ADE.

Follow these steps to develop an NI Switch Executive application in LabVIEW:

1. Open an existing or new LabVIEW VI.
2. Locate the NI Switch Executive VIs. From the **Functions** palette, select **Measurement I/O»Switch Executive**.
3. Select the VIs that you want to use and drop them on the block diagram to build your application.

Example Programs

Use the LabVIEW Example Finder to search or browse examples. NI Switch Executive examples are classified by keyword, so you can search for a particular device or measurement function.

To browse the NI Switch Executive examples available in LabVIEW, launch LabVIEW, click **Help»Find Examples**, and navigate to **Toolkits and Modules»NI Switch Executive**.

For additional information regarding NI Switch Executive examples, refer to [Examples](#).

Using NI Switch Executive in LabWindows/CVI

This topic assumes that you are using the LabWindows/CVI ADE to manage your code development and that you are familiar with the ADE.

Follow these steps to develop an NI Switch Executive application in LabWindows/CVI:

1. Open an existing or new project file.
2. Load the NI Switch Executive function panel(s).
 - (Run-time) `nise.lfp` at `<CVI>\bin` or `\Shared\<CVI>\bin`.
 - (Configuration) `niseCfg.fpf` at `<CVI>\bin` or `\Shared\<CVI>\bin`.
3. Use the function panel to navigate the function hierarchy and generate function calls with the proper syntax and variable values.

Example Programs

For additional information regarding NI Switch Executive examples, refer to [Examples](#).

Using NI Switch Executive in Visual C++

This topic assumes that you are using the Microsoft Visual C++ ADE to manage your code development and that you are familiar with the ADE.

Follow these steps to develop an NI Switch Executive application in Visual C++:

1. Open an existing or new Visual C++ project.
2. Create source files of type C source code (`.c`) or C++ source code (`.cpp`) and add them to the project. Make sure that you include the NI Switch Executive header file, `nise.h`, in your source code files as follows: `#include "nise.h"`.
3. Specify the directory that contains the NI Switch Executive header file under the **Preprocessor»Additional include directories** settings in your compiler—for Visual C++ 6.0 these files are under **Project»Settings»C/C++**. The NI Switch Executive header files are located at `<SwitchExecutive>\API\C`.
4. Add the NI Switch Executive import library `nise.lib` to the project under **Link»General»Object/Library Modules**. The NI Switch Executive import library files are located in the `<SwitchExecutive>\API\C` directory within your NI Switch Executive directory.



Tip The `nise.h` file includes `visatypes.h` from NI-VISA. If the compiler is having trouble locating that file, you can add your VXIPnP include directory (e.g. "`c:\vxipnp\winnt\include`") to the project's list of directories to include from.

5. Add NI Switch Executive function calls to your application.
6. Build your application.

String Passing

To pass strings, pass a pointer to the first element of the character array. Be sure that the string is null-terminated.

Parameter Passing

By default, C passes parameters by value. Remember to pass pointers to variables when you need to pass by address.

Using NI Switch Executive in Visual Basic

This topic assumes that you are using the Microsoft Visual Basic ADE to manage your code development and that you are familiar with the ADE.

Follow these steps to develop an NI Switch Executive application in Visual Basic:

1. Open an existing or new Visual Basic project.
2. Create files necessary for your application: form definition and event handling code (`.frm`), Visual Basic generic code module (`.bas`), or Visual Basic class module (`.cls`). Add these files to the project.
3. Add a reference to the National Instruments Switch Executive Library, which is part of the NI Switch Executive DLL. In Visual Basic 6.0, select the **Project»References** menu option and **NI Switch Executive Version 1.0** or **NI Switch Executive Configuration API Version 1.0**. If you do not see NI Switch Executive listed there, use the Browse button and browse to `system32\nise.dll`.



Note If NI Switch Executive does *not* display in the references dialog box in Visual Basic or if you are trying to use the type library in a language other than Visual Basic, you can add a reference to the `nise.dll` file in your system directory.

4. Use the Object Browser <F2> to find function prototypes and constants.
5. Add NI Switch Executive function calls to your application.
6. Click **Run**.

Example Programs

For additional information regarding NI Switch Executive examples, refer to [Examples](#). To load an example project with Visual Basic 6.0, select **File»Open Project**, then select the .vbp file of your choice.

String Passing

In Visual Basic, variables of data type String do not need special modifications to be passed to NI Switch Executive functions. Visual Basic automatically appends a null character to the end of a string before passing it (by reference, because strings cannot be passed by value in Visual Basic) to a procedure or function.

Parameter Passing

By default, Visual Basic passes parameters by reference. Prepend the `ByVal` keyword if you need to pass by value.

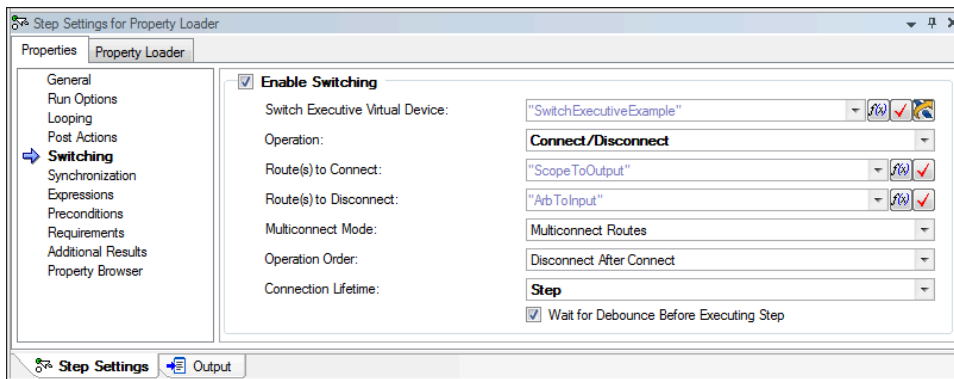
Using NI Switch Executive in NI TestStand

Using NI Switch Executive in NI TestStand, you can configure switching actions for TestStand steps. Implement individual switch operations for your TestStand steps with the Switching panel on the Step Properties dialog box. The IVI Switch step type supports not only the IVI class layer but also NI Switch Executive operations.

TestStand Switching Panel

TestStand includes a Switching panel on the Properties tab of the Step Settings pane. Use this feature with NI Switch Executive.

Use the Switching panel to specify the switching action you want TestStand to perform for a step. The following figure shows the Switching panel:



The Switching panel contains the following controls:

- **Enable Switching**—Specifies whether to perform the specified switching action for the step.
- **Switch Executive Virtual Device**—Specifies an expression that TestStand evaluates at run time to determine the virtual device on which TestStand performs the switching action.
- **Operation**—Specifies whether to connect or disconnect the specified routes or to disconnect all previously connected routes. This operation returns as soon as the instrument is ready for another operation, which may occur before or after the switches involved settle. Select the **Wait For Debounce Before Executing Step** checkbox if you want to wait until all switches have debounced. You can select from the following options:
 - **Connect**—Connects the paths for the specified routes in the **Route(s) to Connect** control.
 - **Disconnect**—Disconnects the paths defined by the specified routes in the **Route(s) to Disconnect** control.
 - **Disconnect All**—Disconnects all paths previously created.
 - **Connect/Disconnect**—Connects the paths for the routes specified in the **Route(s) to Connect** control and disconnects the paths for the routes specified in the **Route(s) to Disconnect** control. Use the **Operation Order** ring control to specify whether the disconnect operation occurs before or after the connect operation.
- **Route(s) to Connect**—Specifies the routes you are connecting. The expression must be a valid route specification string as defined by the NI Switch Executive configuration for the virtual device name you are using. The string can be a combination of route group alias names, route names, and physical route paths.
- **Route(s) to Disconnect**—Specifies the routes you are disconnecting. The expression must be a valid route specification string as defined by the

NI Switch Executive configuration for the virtual device name you are using. The string can be a combination of route group alias names, route names, and physical route paths.

- **Multiconnect Mode**—Specifies the behavior when more than one connection operation occurs on a specific route. You can select from the following options:
 - **Multiconnect Routes**—A route can be connected multiple times. The route must contain the same endpoints and path. NI Switch Executive automatically reference counts the routes. If you issue multiple connect operations for a specific route, the route is not physically disconnected until an equal number of disconnect operations occur. You can either issue the Disconnect operation manually or use a lifetime setting for the route. The Disconnect All operation disconnects a route even if the route reference count is greater than 1.
 - **No Multiconnect**—A route can only be connected once. Any attempt to reconnect a route that is already connected results in an error.
 - **Use Default Setting for Route**—Use the setting defined for the route in the NI Switch Executive virtual device configuration.



Note Use Default Setting for Route was first available in NI Switch Executive 2.0.

- **Operation Order**—Specifies whether the Disconnect operation occurs before or after the Connect operation. You can select from the following options:
 - **Disconnect Before Connect**—Disconnect the specified routes before connecting any routes. Disconnect Before Connect is the typical mode of operation.
 - **Connect Before Disconnect**—Connect the specified routes before disconnecting any routes. Use this mode of operation when you are switching electric current and want to ensure that a load is always connected to your source.
- **Connection Lifetime**—Specifies the lifetime that TestStand applies to the routes specified for the connect or connect/disconnect actions. You can specify whether you want the route to exist until manually disconnected later, or until the step, sequence, thread or execution completes. If you use the Multiconnect Mode, a route can exist longer if another step specifies its own lifetime for the same route. Selecting a lifetime other than manual guarantees that the route stays connected as long as the step, sequence, thread, or execution in which you connect is still executing. If you manually disconnect a route that was previously connected using a non-manual lifetime setting, TestStand releases the reference to the route for the

last step that performed a connect action for that route.

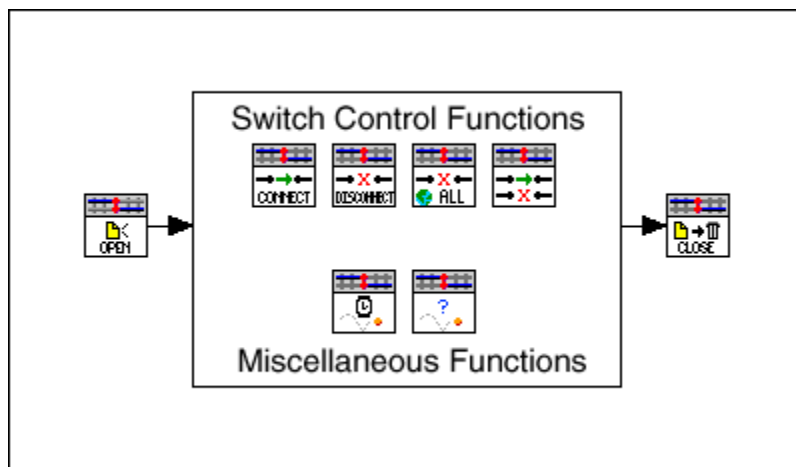
- **Wait for Debounce Before Executing Step**—Specifies whether the operation waits for all switches to debounce before returning to TestStand. The wait for debounce occurs after both Connect and Disconnect operations are complete.

Programming Flow

Complete the following steps to program an NI Switch Executive virtual device:

1. Open a session.
2. Control the virtual device.
3. (Optional) Handle Errors.
4. Close the session.

The following figure represents the programming flow of NI Switch Executive:



Opening a Session

Open a session to communicate with an NI Switch Executive virtual device. This is similar to opening a file before reading/writing to it.

To open a session, use the [niSE Open Session VI](#) or the [niSE_OpenSession](#) function.

Controlling the Virtual Device

Use the following VIs and functions to control an NI Switch Executive virtual device:

- [niSE Connect VI](#) and [niSE_Connect](#) function
- [niSE_ConnectAndDisconnect VI](#) and [niSE_ConnectAndDisconnect](#) function
- [niSE Disconnect VI](#) and [niSE_Disconnect](#) function
- [niSE Disconnect All VI](#) and [niSE_DisconnectAll](#) function
- [niSE Find Route VI](#) and [niSE_FindRoute](#) function

The control VIs and functions accept [route specification strings](#) to designate the path, or paths, the switching system should connect.



Note You must [open a session](#) to the virtual device before you can control it.

Error Handling

Use error handling to assist in troubleshooting problems with switches and debug programming. NI Switch Executive VIs contain `error in` and `error out` nodes to handle errors. NI TestStand run-time errors display when an error has occurred. C and Visual Basic use combinations of the `niSE_GetError` and `niSE_ClearError` functions to report errors. The C and Visual Basic examples contain the function `NI_Switch_ExecutiveCheckErr` that you can use in your own programs.

Refer to [Error Codes](#) for error descriptions.

Closing a Session

Close a session to end communication between the NI Switch Executive virtual device and the software.

To close a session, use the [niSE Close Session VI](#) or the [niSE_CloseSession](#) function.

Route Specification Strings

Route specification strings are the paths connecting two channels and are composed of one or more routes delimited by ampersands (&). For example, in the following line of syntax, there are three defined routes or route groups:

```
routeOrGroup & routeOrGroup & routeOrGroup...
```

where `routeOrGroup` can be the following:

- Route name
- Route group name
- Two endpoint channels to be connected that are delimited by `->`. NI Switch Executive dynamically determines the path between the endpoints. In this mode, a hardware alias name can be substituted for the endpoints.

```
channel -> channel
```

- Fully specified path enclosed in square brackets consisting of one or more endpoint channels delimited by `->`:

```
[channel -> channel -> channel...]
```

where `channel` can be the following:

- A channel alias name
- A unique name created by combining the IVI device logical name and IVI channel name separated by a forward slash (/) delimiter, such as `device/iviChan`.



Tip NI Switch Executive has the option of finding an available path at run time by slightly modifying the syntax. Remove the square brackets and enter the two endpoints of a route delimited by a `->`.

Note



- Any `channel`, other than an endpoint, within a route specification string **must** be reserved for routing or directly hardwired to one of the endpoint channels.
- `channels` used as endpoints must **not** be reserved for routing channels.
- When connecting a route, the list of channels must obey the exclusion rules both explicitly in the route specification string as well as implicitly by any previous connections. Exclusion violations result in an error.

The following are samples of route specification strings for a matrix:

Route Specification String Examples	Description
A->B	Connect channel A to channel B. NI Switch Executive automatically determines the path between the two channels.
[SampleMatrix1/c0->SampleMatrix1/r1->SampleMatrix1/c4]	A fully specified path between column 0 and column 4 of SampleMatrix1. Row 1 must be marked as reserved for routing.
[Scope->R3->SampleMatrix1/c6]	A fully specified path between the channel named Scope and column 6 of SampleMatrix1. Row 3 must be marked as reserved for routing.
Scope->SampleMatrix1/c6	Connect the channel named Scope to column 6 of SampleMatrix1. NI Switch Executive automatically determines the path between the two channels.
ArbToInput & ScopeToOutput	Complete the connections for route group ArbToInput and ScopeToOutput.
PowerDevice & [Scope->R3->UUT_Out]	Complete the connections for route group PowerDevice and complete the fully specified path between the channel named Scope and the channel named UUT_Out. R3 must be marked as a reserved for routing.

Each supported ADE has an associated Route Specification String Example. Refer to the [Examples](#) for the appropriate ADE to see how route specification strings are used when programming.

Session Reference Counting

NI Switch Executive uses a reference counting scheme to manage open session handles to an NI Switch Executive virtual device. Each call to the [niSE Open Session VI](#) or the [niSE_OpenSession](#) function must be matched with a subsequent call to the [niSE Close Session VI](#) or the [niSE_CloseSession](#) function.

Multiple calls to the [niSE Open Session VI](#) or the [niSE_OpenSession](#) function with the same virtual device name always return the same session handle. Only after all session handles are closed to a given virtual device can NI Switch Executive disconnect its communication with the IVI switches. Session handles can be used safely in multiple threads of an application.



Note When using an NI Switch Executive virtual device in an ADE, always allow the application to complete its call to the niSE Close Session VI or the `niSE_CloseSession` function. If you abort or halt the execution of your program before calling the niSE Close Session VI or the `niSE_CloseSession` function, the application may leave the session open. When a session remains open, all previous connections continue to exist.



Tip If you are debugging and prefer not to call the niSE Close Session VI or the `niSE_CloseSession` function, you can call the `niSE_DisconnectAll` VI or `niSE_DisconnectAll` function immediately after calling the niSE Open Session VI or the `niSE_OpenSession` function to ensure a known startup state.

Specification Type

Specification type is the connection method the routing engine uses to determine a route.

On the **Routes/Groups** tab, select from the following connection methods:

- **Specified By Full Path**—The routing engine uses the fully specified path to determine the route.
- **Find By Endpoints**—The routing engine uses endpoints to determine the route.
- **Find If Full Path Fails**—If the full path fails to connect, the routing engine uses endpoints to determine the route.

Multiconnect Mode

The typical state of a switch is either connected or disconnected. Physically, there is no way for a switch to be connected more than once—it is either connected or it is not. In certain situations, however, it is useful to think of a connection as something that can happen more than once.

For example, consider having two tests that can operate simultaneously. Although these tests primarily use different instruments and test points, they share a connection from a power supply to the UUT power input. Instead of having to manually handle this shared connection by some extra component, it is preferable to create the two tests independently such that they could be operated individually or simultaneously. NI Switch Executive facilitates this sort of sharing with multiconnection.

You can make a connection with ***no multiconnection*** or ***multiconnect routes***.

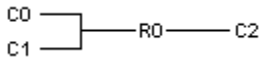
If a route is connected with no multiconnection, the route must be disconnected before it can be reconnected. If you try to connect a route that was connected without multiconnection, an error condition occurs.

If a route is connected with multiconnect routes, the route can be connected multiple times. The first connection call performs a physical hardware connection. Successive calls merely increase a reference count on the connection. Similarly, disconnect calls decrease the reference count. When the reference count is at 0, the hardware is physically disconnected.

When you [add a route](#), you can set the default multiconnect mode for the route—**Multiconnect Routes** or **No Multiconnect**. If you do not set the default multiconnect mode, the mode defaults to **Multiconnect Routes**.

Sharing Route Legs

It is possible to share route legs as long as the routes share a common start or endpoint as shown in the following figure:



NI Switch Executive manages the legs internally, keeping the leg connected until it is no longer needed.

Using Properties and Attributes

NI Switch Executive contains high-level VIs and functions that set most of the device properties and attributes. Using the configuration API, you can use additional properties and attributes to modify NI Switch Executive virtual device configurations.

Refer to [NI Switch Executive LabVIEW Reference](#) or [NI Switch Executive C Function Reference](#) for a complete listing of the available properties and attributes in NI Switch Executive.

Accessing Properties

In LabVIEW, properties are accessed through the Switch Executive property node.

Complete the following steps to access properties in LabVIEW:

1. Open a VI.
2. Right-click in the block diagram view to open the **Functions** palette.
3. Select **Measurement I/O»Switch Executive»Configuration»Property Nodes** to open the Switch Executive Configuration API Property Nodes palette.
4. Drag and drop the icon of the specific property node you want to use to the block diagram.
5. Left-click the property node and select the property that you want to use.
6. Resize the property node to add additional properties by dragging the resizing handle at the top or bottom of the node and release the mouse button.

Setting Properties and Attributes Before Reading Them

Properties and attributes are modified when you set them or when you call a configuration VI or function that sets them, respectively. It is important to set the properties or attributes or call any configuration VIs or functions before reading back any property or attribute values for the following reasons:

- Values read are coerced depending on the current configuration of the session. If you read a property or attribute value and then set other properties or attributes, the value read may no longer be valid.
- The driver verifies that the configuration of the device is valid at the time the property or attribute is read. It is possible to get an error when reading a property or attribute if the configuration is not valid at that point, even when a setting later could make it valid.
- Reading properties or attributes causes the driver to verify the current configuration. If you change some of the settings later, those settings need to be validated again.



Note Perform all module configuration before writing data on output devices.

Debugging with NI I/O Trace

NI I/O Trace is an application that monitors, records, and displays National Instruments API calls made by applications. Use NI I/O Trace to quickly locate and analyze any erroneous National Instruments API calls that NI Switch Executive makes and to verify that the communication with your instrument is correct.

To launch NI I/O Trace, navigate to **Start»All Programs»National Instruments»NI IO Trace**.

To begin monitoring or capturing API calls, press <F8>. To end capture, press <Ctrl-Break>.



Tip Double-click a function in NI I/O Trace, or right-click and choose **Properties**, to display a dialog box with more information about the inputs and outputs of the function call.

NI I/O Trace Monitoring NI Switch Executive Function Calls

The screenshot shows the NI I/O Trace application window with the title bar "NI I/O Trace - [capture on]". The window contains a menu bar (File, Edit, View, Tools, Help) and a toolbar with icons for file operations and execution. Below the toolbar is a table with the following data:

Number	Description	Status	Time
1	niSE_OpenSession ("VirtualDevice1", "", VirtualDevice1)	VI_SUCCESS	07:51:36.283
2	niSE_GetAllConnections (VirtualDevice1, 0x1C8465D0, 0x1D824B20)	VI_SUCCESS	07:51:36.513
3	niSE_Connect (VirtualDevice1, "RouteGroup0", -1, VI_TRUE)	VI_SUCCESS	07:51:45.828
4	niSE_Connect (VirtualDevice1, "Route0", -1, VI_TRUE)	VI_SUCCESS	07:51:51.355
5	niSE_Connect (VirtualDevice1, "Route1", -1, VI_TRUE)	0xFFFF8E98	07:52:00.493
6	niSE_GetError (VirtualDevice1, NULL, NULL, 0x1D80F990)	VI_SUCCESS	07:52:00.493
7	niSE_GetError (VirtualDevice1, 0x1D80F9E4, "Connecting this...ls:PXI1Slot2)	VI_SUCCESS	07:52:00.493
8	niSE_Disconnect (VirtualDevice1, "Route0")	VI_SUCCESS	07:52:17.597
9	niSE_CloseSession (0x0F3754F8)	VI_SUCCESS	07:52:18.890

At the bottom of the window, there are two input fields: "Captured" with the value 9 and "Displayed" with the value 9.



Tip NI I/O Trace supports two buffer sizes. The default buffer size, brief, only allows 64 characters in any of the string fields. To view full error descriptions and string inputs and outputs, NI recommends using the full buffer option.

For more information about using NI I/O Trace, refer to the ***NI I/O Trace Help***, available in NI I/O Trace at **Help»Help Topics**.

NI Switch Executive LabVIEW Reference

Expand this topic to view the VIs and properties that you can use to configure and operate an NI Switch Executive virtual device.

NI Switch Executive VIs

August 2015, 372552D-01

Use the VIs on the NI Switch Executive palette to build the block diagram.

Palette Object	Description
niSE Open Session VI	Opens a session to a specified NI Switch Executive virtual device. Opens communications with all of the IVI switches associated with the specified NI Switch Executive virtual device. Returns a session handle that you use to identify the virtual device in all subsequent NI Switch Executive VI calls.
niSE Close Session VI	Reduces the reference count of open sessions by one. If the reference count goes to zero, the VI closes any open IVI switch sessions.
niSE Connect VI	Connects the routes specified by the connection specification. When connecting, it may allow for multiconnection based on the multiconnection mode.
niSE Connect And Disconnect VI	<p>Connects the routes specified by the Connection Specification. When connecting, it may allow for multiconnection based on the Multiconnect Mode. Disconnects the routes specified in the Disconnection Specification. This VI is useful for switching from one state to another state.</p> <p>This function will not wait for debounce. Call the niSE Wait for Debounce function to wait for debounce.</p>

<u>niSE Disconnect VI</u>	<p>Disconnects the routes specified in the Disconnection Specification parameter.</p> <p>This function will not wait for debounce. Call the niSE Wait for Debounce function to wait for debounce.</p>
<u>niSE Disconnect All VI</u>	<p>Disconnects all connections on every IVI switch device managed by the NI Switch Executive session reference passed to this VI.</p> <p>This function will not wait for debounce. Call the niSE Wait for Debounce function to wait for debounce.</p>
<u>niSE Expand Route Spec VI</u>	<p>Expands a route spec string to yield more information about the routes and route groups within the spec.</p>
<u>niSE Find Route VI</u>	<p>Finds an existing or potential route between channel 1 and channel 2.</p>
<u>niSE Get All Connections VI</u>	<p>Returns all connected routes and route groups.</p>
<u>niSE Is Connected VI</u>	<p>Checks to see whether the specified route or route group is connected.</p>
<u>niSE Is Debounced VI</u>	<p>Checks to see if the switching system is debounced or not. This VI does not wait for debouncing to occur. It returns true if the system is fully debounced.</p> <p>This VI is similar to the lviSwch specific function.</p>
<u>niSE Wait For Debounce VI</u>	<p>Waits for all of the switches in the NI Switch Executive virtual device to debounce.</p>

Subpalette	Description
------------	-------------

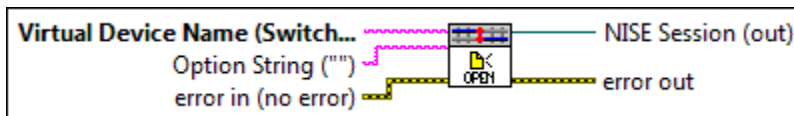
<p><u>Configuration</u></p>	<p>Use the VIs on the Configuration subpalette to programmatically access the NI Switch Executive Virtual Device configuration.</p>
<p><u>Low Level</u></p>	<p>Use the VIs on the IVI and NI-SWITCH subpalettes to access IVI sessions to instrument drivers of switch devices that are a part of a session to the Switch Executive Virtual Device.</p>




© 2002–2015 National Instruments. All Rights Reserved.















niSE Open Session VI

Opens a session to a specified NI Switch Executive virtual device. Opens communications with all of the IVI switches associated with the specified NI Switch Executive virtual device. Returns a session handle that you use to identify the virtual device in all subsequent NI Switch Executive VI calls.

Details



	<p>Virtual Device Name (SwitchConfiguration1) is the session referencing this NI Switch Executive virtual device.</p>
	<p>Option String (\"") can be used to pass information to each of the IVI devices on startup. It can be used to set things such as simulation, range checking, and so on. Consult your driver documentation for more information about valid entries for the option string.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1140 1469 1663"> <tr> <td data-bbox="240 1140 321 1308">  </td> <td data-bbox="329 1140 1469 1308"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1318 321 1486">  </td> <td data-bbox="329 1318 1469 1486"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1497 321 1663">  </td> <td data-bbox="329 1497 1469 1663"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

Details

NI Switch Executive uses a reference counting scheme to manage open session handles to an NI Switch Executive virtual device. Each call to niSE Open Session must be matched with a subsequent call to the [niSE Close Session VI](#).

Successive calls to the niSE Open Session VI with the same virtual device name always returns the same session handle.

NI Switch Executive disconnects its communication with the IVI switches after all session handles are closed to a given virtual device.

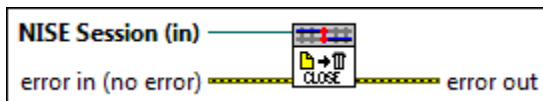
The session handles can be used safely in multiple threads of an application.

Sessions can only be opened to a given NI Switch Executive virtual device from a single process at a time.






niSE Close Session VI

Reduces the reference count of open sessions by one. If the reference count goes to zero, the VI closes any open IVI switch sessions.

Details



	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p>

	indicate a warning or that no error occurred before this VI ran. The default is FALSE.
	code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
	error out contains error information. This output provides standard error out functionality .
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

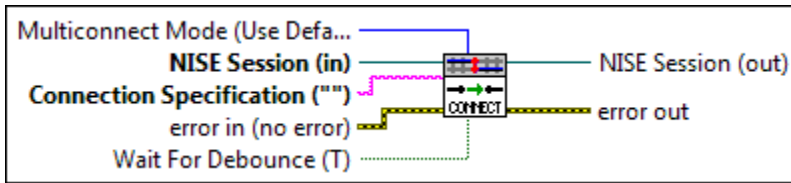
Details

After calling the niSE Close Session VI, you should not use the NI Switch Executive virtual device again until you call the [niSE Open Session VI](#).

niSE Connect VI

Connects the routes specified by the connection specification. When connecting, it may allow for multiconnection based on the multiconnection mode.

Details






Multiconnect Mode (Use Default Mode) sets the connection mode for the VI.

The mode may be one of the following:

<p>Default (-1)</p>	<p>Uses the mode selected as the default for the route in the NI Switch Executive virtual device configuration. If a mode has not been selected for the route in the NI Switch Executive virtual device, this parameter defaults to <code>Multiconnect Routes</code>.</p>
<p>No Multiconnect (0)</p>	<p>Routes specified in the connection specification must be disconnected before they can be reconnected. Calling the niSE Connect VI on a route that was connected using No Multiconnect mode results in an error condition.</p>
<p>Multiconnect Routes (1)</p>	<p>Routes specified in the connection specification can be connected multiple times. The first call to the niSE Connect VI performs the physical hardware connection. Successive calls to the niSE Connect VI increase a connection reference count. Similarly, calls to the niSE Disconnect VI decrease the reference count. When it reaches 0, the hardware is physically disconnected. This behavior is slightly different with SPDT relays. For more information, refer to Exclusions and SPDT Relays. Multiconnecting routes applies to entire routes and not to route segments.</p>

NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the [niSE Open Session VI](#).

	<p>Connection Specification ("") is the string describing the connections to be made. Refer to Route Specification Strings for more information.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Wait For Debounce (T), if TRUE, waits for all switches to debounce before it returns. If FALSE, it immediately returns after the switch control is completed.</p>
	<p>Note In either case, the wait for debounce occurs after the connection portion of operation. If you need finer grained control of this debounce behavior, consider calling the niSE Connect VI and the niSE Disconnect VI separately.</p>
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>

	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

Details

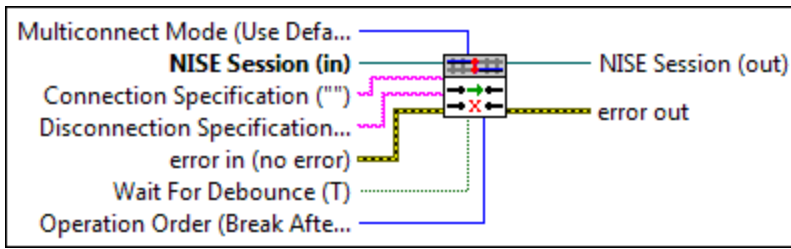
In the event of an error, the call to the niSE Connect VI attempts to undo any connections made so that the system is left in the same state that it was in before the call was made. Some errors can be caught before manipulating hardware, although it is feasible that a hardware call could fail causing some connections to be momentarily closed and then reopened.

niSE Connect And Disconnect VI

Connects the routes specified by the **Connection Specification**. When connecting, it may allow for multiconnection based on the **Multiconnect Mode**. Disconnects the routes specified in the **Disconnection Specification**. This VI is useful for switching from one state to another state.

This function will not wait for debounce. Call the niSE Wait for Debounce function to wait for debounce.

Details



Multiconnect Mode (Use Default Mode) sets the connection mode for the VI.

The mode may be one of the following:

<p>Default (-1)</p>	<p>Uses the mode selected as the default for the route in the NI Switch Executive virtual device configuration. If a mode has not been selected for the route in the NI Switch Executive virtual device, this parameter defaults to <code>Multiconnect Routes</code>.</p>
<p>No Multiconnect (0)</p>	<p>Routes specified in the connection specification must be disconnected before they can be reconnected. Calling the <code>niSE Connect VI</code> on a route that was connected using No Multiconnect mode results in an error condition.</p>
<p>Multiconnect Routes (1)</p>	<p>Routes specified in the connection specification can be connected multiple times. The first call to the <code>niSE Connect VI</code> performs the physical hardware connection. Successive calls to the <code>niSE Connect VI</code> increase a connection reference count. Similarly, calls to the <code>niSE Disconnect VI</code> decrease the reference count. When it reaches 0, the hardware is physically disconnected. This behavior is slightly different with SPDT relays. For more information, refer to Exclusions and SPDT Relays. Multiconnecting routes applies to entire routes and not to route segments.</p>

NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the [niSE Open Session VI](#).

	<p>Connection Specification (""") is the string describing the connections to be made. Refer to Route Specification Strings for more information.</p>
	<p>Disconnection Specification (""") is the string describing the disconnections to be made. Refer to Route Specification Strings for more information.</p> <p>You can use an asterisk in this parameter to accomplish the following:</p> <ol style="list-style-type: none"> 1. For NISE_VAL_BREAK_BEFORE_MAKE operation order, an asterisk (*) will disconnect all connections currently connected and connect what is in the Connection Specification parameter. However, this action maintains as closed those connections currently connected that also exist in the Connection Specification parameter. This behavior minimizes relay actuation and thus extends the life of the relays. 2. For NISE_VAL_BREAK_AFTER_MAKE operation order, the VI connects what is in the Connection Specification parameter and the asterisk (*) will disconnect all other connections. As in case 1, this action maintains as closed those connections currently connected that also exist in the Connection Specification parameter. This behavior minimizes relay actuation and thus extends the life of the relays.
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

Wait For Debounce (T), if TRUE, waits for all switches to debounce before it returns. If FALSE, it immediately returns after the switch control is completed.



Note In either case, the wait for debounce occurs after the connection portion of operation. If you need finer grained control of this debounce behavior, consider calling the [niSE Connect VI](#) and the [niSE Disconnect VI](#) separately.

Operation Order (Break After Make) sets the order of the operation for the VI. Defined values are Break Before Make and Break After Make.



<p>Break Before Make (1)</p>	<p>The VI disconnects the routes specified in the Disconnection Specification before connecting the routes specified in the Connection Specification. Break Before Make is the typical mode of operation.</p>
<p>Break After Make (2)</p>	<p>The VI connects the routes specified in the Connection Specification before connecting the routes specified in the Disconnection Specification. This mode of operation is normally used when you are switching current and want to ensure that a load is always connected to your source. The order of operation is to connect first or disconnect first.</p>





NI SE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the [niSE Open Session VI](#). This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.

error out contains error information. This output provides standard [error out functionality](#).



status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

Details

The niSE Connect And Disconnect VI manipulates the hardware connections and disconnections only when the routes are different between the connection and disconnection specifications. If any routes are common between the connection and disconnection specifications, NI Switch Executive determines whether the relays need to be switched. This functionality has the distinct advantage of increased throughput for shared connections, because hardware does not have to be involved and potentially increases relay lifetime by decreasing the number of times that the relay has to be switched.

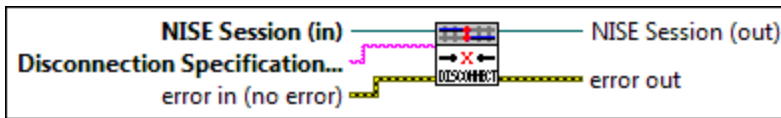
In the event of an error, the call to the niSE Connect And Disconnect VI attempts to undo any connections made, but does not attempt to reconnect disconnections. Some errors can be caught before manipulating hardware, although it is feasible that a hardware call could fail causing some connections to be momentarily closed and then reopened.

niSE Disconnect VI




Disconnects the routes specified in the **Disconnection Specification** parameter.

This function will not wait for debounce. Call the niSE Wait for Debounce function to wait for debounce.

Details



	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
	<p>Disconnection Specification ("") is the string describing the disconnections to be made. Refer to Route Specification Strings for more information.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>

	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

Details

If any of the specified routes were originally connected in a multiconnected mode, the call to the niSE Disconnect VI reduces the reference count on the route by 1. If the reference count reaches 0, it is disconnected.

If a specified route does not exist, it is an error condition.

In the event of an error, the call to the niSE Disconnect VI continues to try to disconnect everything specified by the route specification string but reports the error on completion.

niSE Disconnect All VI



Disconnects all connections on every IVI switch device managed by the NI Switch Executive session reference passed to this VI.

This function will not wait for debounce. Call the niSE Wait for Debounce function to wait for debounce.

Details



	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>

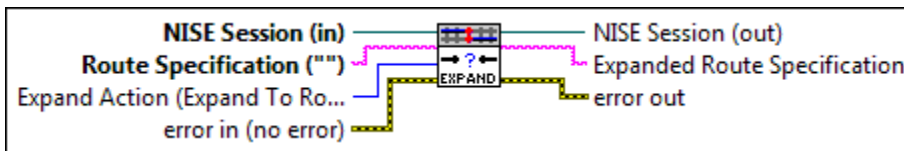
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>




Details






The niSE Disconnect All VI ignores all multiconnect modes. Calling the niSE Disconnect All VI resets all of the switch states for the system.

niSE Expand Route Spec VI




Expands a route spec string to yield more information about the routes and route groups within the spec.



	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
	<p>Route Specification (\"") is the string describing the routes and route groups to be expanded. Refer to Route Specification Strings for more information.</p>
	<p>Expand Action (Expand To Routes)</p> <p>The mode can be one of the following:</p>

Expand to Routes (0)	expands the route spec to routes. Converts route groups to their constituent routes.
Expand to Paths (1)	expands the route spec to paths. Converts routes and route groups to their constituent square bracket route spec strings. Example: [Dev1/c0->Dev1/r0->Dev1/c1]
<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>	
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>
	<p>Expanded Route Specification ("") is the string containing the results of the route and route group expansion. Refer to Route Specification Strings for more information.</p>

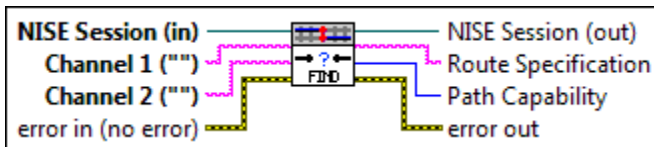
error out contains error information. This output provides standard [error out functionality](#).




	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niSE Find Route VI




Finds an existing or potential route between channel 1 and channel 2.

Details



	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
	<p>Channel 1 (\"'\") is the channel name of one of the endpoints of the route to find. The channel name must either be a channel alias name or a name in the <code>device/ivichannel</code> syntax.</p>
	<p>Channel 2 (\"'\") is the channel name of one of the endpoints of the route to find. The channel name must either be a channel alias name or a name in the <code>device/ivichannel</code> syntax.</p>

error in (no error) describes error conditions that occur before this node runs. This input provides standard [error in functionality](#).

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the [niSE Open Session VI](#). This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.

Route Specification contains the fully specified route path surround by square brackets—if the route exists or is possible. Refer to [Route Specification Strings](#) for more information.




Path Capability is the return value which expresses the capability of finding a valid route between channel 1 and channel 2.

<p>Path Available (1)</p>	<p>A path between channel 1 and channel 2 is available. The Route Specification parameter returns a string describing the available path.</p>
<p>Path Exists (2)</p>	<p>A path between channel 1 and channel 2 already exists. The Route Specification parameter returns a string describing the existing path.</p>

Path Unsupported (3)	There is no potential path between channel 1 and channel 2 given the current configuration.
Resource In Use (4)	There is a potential path between channel 1 and channel 2, although a resource needed to complete the path is already in use.
Exclusion Conflict (5)	Channel 1 and channel 2 cannot be connected because their connection would result in an exclusion violation.
Channel Not Available (6)	One of the channels is not useable as an endpoint channel. Make sure that it is not marked as a configuration channel.
Channels Hardwired (7)	The two channels reside on the same hardware. An implicit path already exists.

error out contains error information. This output provides standard [error out functionality](#).



	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

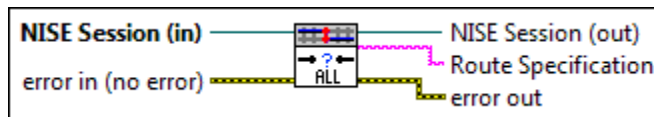
Details

The returned route specification contains the route specification, and the route capability indicates whether the route existed, is possible, or is not possible for various reasons.

The **Route Specification** string returned from the niSE Find Route VI can be passed to other NI Switch Executive API VIs, such as the [niSE Connect VI](#), [niSE Disconnect VI](#), and [niSE Connect And Disconnect VI](#), that use [Route Specification Strings](#).

niSE Get All Connections VI

Returns all connected routes and route groups.

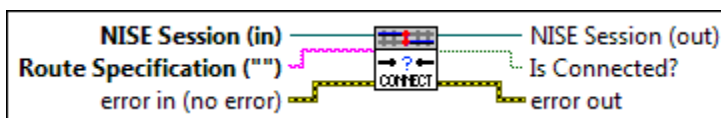


	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>
	<p>Route Specification ("") is the string describing the connected routes and route groups. Refer to Route Specification Strings for more information.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>



niSE Is Connected VI

Checks to see whether the specified route or route group is connected.



	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

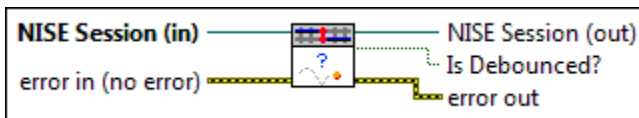
	<p>Route Specification ("") is the string describing the routes and route groups to be checked. Refer to Route Specification Strings for more information.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>NI SE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>
	<p>Is Connected? returns TRUE if the routes and route groups are connected or FALSE if they are not connected.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>









	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>














niSE Is Debounced VI

Checks to see if the switching system is debounced or not. This VI does not wait for debouncing to occur. It returns true if the system is fully debounced.

This VI is similar to the IviSwch specific function.



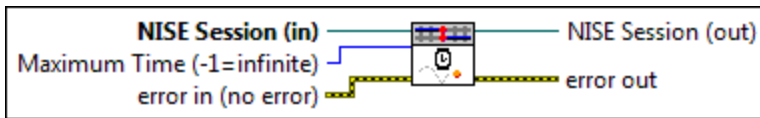
	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>				
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1457 1468 1801"> <tr> <td data-bbox="250 1457 321 1625">  </td> <td data-bbox="321 1457 1468 1625"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="250 1625 321 1793">  </td> <td data-bbox="321 1625 1468 1793"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>				
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>				

	<p> source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>						
	<p>Is Debounced? returns TRUE if the system is fully debounced or FALSE if it is still settling.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 926 1468 1444"> <tr> <td data-bbox="240 926 321 1100">  </td> <td data-bbox="321 926 1468 1100"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1100 321 1274">  </td> <td data-bbox="321 1100 1468 1274"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1274 321 1444">  </td> <td data-bbox="321 1274 1468 1444"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						




niSE Wait For Debounce VI

Waits for all of the switches in the NI Switch Executive virtual device to debounce.

[Details](#)



	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>
	<p>Maximum Time (-1=infinite) is the amount of time to wait (in milliseconds) for the debounce to complete. A value of 0 checks for debouncing once and returns an error if the system is not debounced at that time. A value of -1 means to wait for an infinite period of time until the system is debounced.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>

	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

Details

This VI does not return until either the switching system is completely debounced and settled or the maximum time has elapsed and the system is not yet debounced.

In the event that the maximum time elapses before the relay has switched and debounced, the VI returns an error indicating that a timeout has occurred.

To ensure that all of the switches have settled, NI recommends calling the niSE Wait For Debounce VI after a series of connection or disconnection operations and before taking any measurements of the signals connected to the switching system.



Note If you use Wait for Debounce on the [niSE Connect And Disconnect VI](#), calling the niSE Wait for Debounce VI again is not necessary.

Configuration

Owning Palette: [NI Switch Executive VIs](#)

Use the VIs on the Configuration subpalette to programmatically access the NI Switch Executive Virtual Device configuration.

Palette Object	Description
----------------	-------------

<u>niseCfg Open Configuration VI</u>	Opens a session to the NI Switch Executive configuration on your system.
<u>niseCfg Save VI</u>	Saves the changes made to all NI Switch Executive virtual device configurations during the session.
<u>niseCfg Close Configuration VI</u>	Closes the NI Switch Executive configuration session.
<u>niseCfg Get Object VI</u>	Returns the reference to an object.
<u>niseCfg Get Objects VI</u>	Returns the reference to an object.
<u>niseCfg Close Object VI</u>	Closes an object in the configuration.
<u>niseCfg Add Object VI</u>	Adds an object to the collection.
<u>niseCfg Remove Object VI</u>	Removes an object. After NI Switch Executive removes an object, it does not need to be released using the <u>niseCfg Close Object VI</u> .
<u>niseCfg Import VI</u>	Imports a virtual device configuration from an NI Switch Executive configuration file.
<u>niseCfg Export VI</u>	Exports the virtual device configuration to an NI Switch Executive Export File.
<u>niseCfg Import Specific VI</u>	Imports a virtual device configuration from a file.

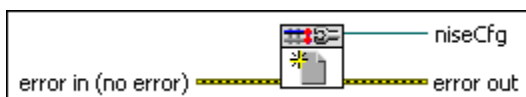
niseCfg Export Previous VI	Exports the virtual device configuration to a file format coinciding with a previous version of NI Switch Executive.
niseCfg Export Specific VI	Exports the virtual device configuration to a specified file format.

Subpalette	Description
Property Nodes	Use the NI Switch Executive property nodes to access properties of individual objects of the Switch Executive Virtual Device.








niseCfg Open Configuration VI

Opens a session to the NI Switch Executive configuration on your system.

This VI returns a session handle. Use the session handle to obtain references to individual NI Switch Executive virtual device configurations and perform operations on the NI Switch Executive configuration such as Add New Virtual Device and Save. Use the [niseCfg Close Configuration VI](#) to close the session when complete. Unlike the session created with the niSE Open Session VI, this session is not a run-time session.

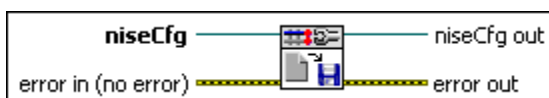


<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>	
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>

	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
<p> niseCfg is the NI Switch Executive configuration session handle.</p>	
<p>error out contains error information. This output provides standard error out functionality.</p>	
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p> code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseCfg Save VI

Saves the changes made to all NI Switch Executive virtual device configurations during the session.



	<p>niseCfg is the NI Switch Executive configuration session handle. Call the niseCfg Open Configuration VI to obtain the session handle.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>niseCfg out is the NI Switch Executive configuration session handle. Call the niseCfg Open Configuration VI to obtain the session handle.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of</p>

the VI that produced the error or warning.

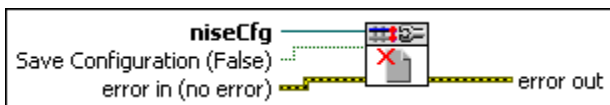
niseCfg Close Configuration VI

Closes the NI Switch Executive configuration session.

Specify whether to save changes to the disk with the **Save Configuration** parameter. If you have made changes to the configuration since the last call to Save and you want to save those changes, pass True for the **Save Configuration** parameter. If you have made changes that you want to revert, pass False for the **Save Configuration** parameter. If you have not made changes to the configuration (for example, you have only browsed the object properties), closing the configuration does not affect the saved data.



Note Import Virtual Device implicitly saves the configuration. All changes up to that point are saved.









niseCfg is the NI Switch Executive configuration session handle. Call the [niseCfg Open Configuration VI](#) to obtain the session handle.



Save Configuration (False) specifies whether to save the configuration. Pass True to save to the configuration. Pass False to close the NI Switch Executive configuration session without saving.



error in (no error) describes error conditions that occur before this node runs. This input provides standard [error in functionality](#).

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
<p>error out contains error information. This output provides standard error out functionality.</p>	
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

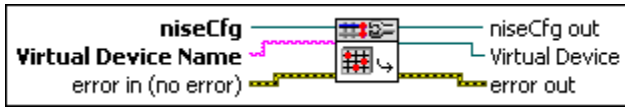
niseCfg Get Object VI

Returns the reference to an object.

All objects that you retrieve from the API must be closed with the [niseCfg Close Object VI](#).




niseCfg Get Virtual Device

Returns the virtual device from the NI Switch Executive configuration.



	niseCfg is the NI Switch Executive configuration session handle. Call the niseCfg Open Configuration VI to obtain the session handle.
	Virtual Device Name specifies the name of the virtual device to retrieve from the configuration.
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .
	status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
	code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
	niseCfg out is the NI Switch Executive configuration session handle. Call the niseCfg Open Configuration VI to obtain the session handle.
	Virtual Device is the reference to the virtual device.







error out contains error information. This output provides standard [error out functionality](#).















	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Get Bus

Returns a bus on the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>		
	<p>Bus Name specifies the name of the bus to retrieve from the virtual device.</p>		
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1696 1468 1829"> <tr> <td data-bbox="240 1696 321 1829">  </td> <td data-bbox="321 1696 1468 1829"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p>		

	<p>indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> <p> code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> <p> source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Bus is the reference to the bus.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1157 1474 1675"> <tr> <td data-bbox="250 1230 315 1262">  </td> <td data-bbox="315 1157 1474 1329"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="250 1402 315 1434">  </td> <td data-bbox="315 1329 1474 1501"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="250 1570 315 1602">  </td> <td data-bbox="315 1501 1474 1675"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						




niseVirtualDevice Get Channel

Returns a channel on the virtual device.



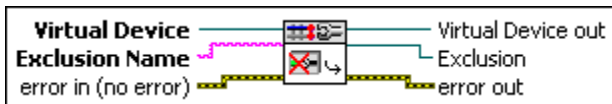
	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Channel Name specifies the alias name or the full channel name of the channel to retrieve from the virtual device.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Channel is the reference to the channel.</p>







error out contains error information. This output provides standard [error out functionality](#).















	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Get Exclusion

Returns the exclusion for the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>		
	<p>Exclusion Name specifies the name of the exclusion to retrieve from the virtual device.</p>		
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1701 1469 1833"> <tr> <td data-bbox="240 1701 321 1833">  </td> <td data-bbox="321 1701 1469 1833"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p>		

	<p>indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Exclusion is the reference to the exclusion.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1150 1468 1671"> <tr> <td data-bbox="240 1150 321 1325">  </td> <td data-bbox="321 1150 1468 1325"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1325 321 1499">  </td> <td data-bbox="321 1325 1468 1499"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1499 321 1671">  </td> <td data-bbox="321 1499 1468 1671"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						




niseVirtualDevice Get Hardware

Returns a hardware on the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Hardware Name specifies the name of the hardware to retrieve from the virtual device.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Hardware is the reference to the hardware.</p>







error out contains error information. This output provides standard [error out functionality](#).
















	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Get IVI Device

Returns an IVI device on the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>		
	<p>IVI Logical Name specifies the IVI Logical Name for the switch device you want to use in the configuration.</p> <table border="1" data-bbox="235 1564 1469 1638"> <tr> <td data-bbox="235 1564 373 1638">  </td> <td data-bbox="373 1564 1469 1638"> <p>Note IVI logical names are case sensitive.</p> </td> </tr> </table>		<p>Note IVI logical names are case sensitive.</p>
	<p>Note IVI logical names are case sensitive.</p>		
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>		

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>IVI Device is the reference to the IVI device.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1192 1468 1713"> <tr> <td data-bbox="240 1192 321 1367">  </td> <td data-bbox="321 1192 1468 1367"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1367 321 1541">  </td> <td data-bbox="321 1367 1468 1541"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1541 321 1713">  </td> <td data-bbox="321 1541 1468 1713"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						




niseVirtualDevice Get Route Group

Returns a route group on the virtual device.



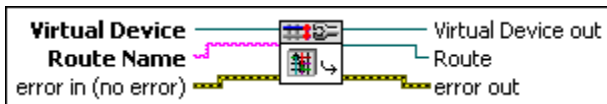
	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Route Group Name specifies the name of the route group you want to retrieve from the virtual device.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Route Group is the reference to the route group.</p>







error out contains error information. This output provides standard [error out functionality](#).








	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Get Route

Returns a route on the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>		
	<p>Route Name specifies the name of the route you want to retrieve from the virtual device.</p>		
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="232 1701 1469 1837"> <tr> <td data-bbox="232 1701 324 1837">  </td> <td data-bbox="324 1701 1469 1837"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to</p>		

	<p>indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Route is the reference to the route.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1150 1468 1671"> <tr> <td data-bbox="240 1150 321 1325">  </td> <td data-bbox="321 1150 1468 1325"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1325 321 1499">  </td> <td data-bbox="321 1325 1468 1499"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1499 321 1671">  </td> <td data-bbox="321 1499 1468 1671"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

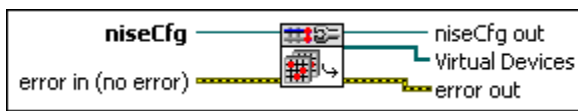
niseCfg Get Objects VI

Returns the reference to an object.











All objects that you retrieve from the API must be closed with the [niseCfg Close Object VI](#).

niseCfg Get Virtual Devices

Returns an array of virtual devices from the NI Switch Executive configuration.

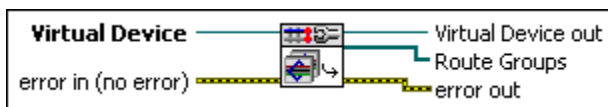




	niseCfg is the NI Switch Executive configuration session handle. Call the niseCfg Open Configuration VI to obtain the session handle.
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .
	status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
	code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
	niseCfg out is the NI Switch Executive configuration session handle. Call the niseCfg Open
















	<p>Configuration VI to obtain the session handle.</p>						
	<p>Virtual Devices is the array of references to the virtual device.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 533 1469 1052"> <tr> <td></td> <td> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td></td> <td> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td></td> <td> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseVirtualDevice Get Route Groups

Returns an array of the route groups on the virtual device.

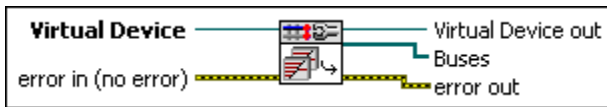


	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>




	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Route Groups is the array of references to the route groups on the virtual device.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1192 1468 1713"> <tr> <td data-bbox="240 1192 321 1367">  </td> <td data-bbox="321 1192 1468 1367"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1367 321 1541">  </td> <td data-bbox="321 1367 1468 1541"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1541 321 1713">  </td> <td data-bbox="321 1541 1468 1713"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseVirtualDevice Get Buses

Returns an array of buses on the virtual device.

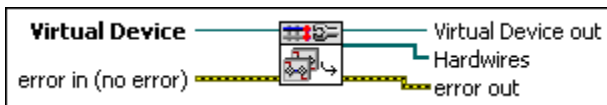










	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Buses is the array of references to the buses on the virtual device.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>














	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Get Hardwires

Returns an array of hardwires on the virtual device.

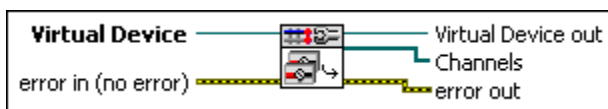


	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>				
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1480 1469 1829"> <tr> <td data-bbox="240 1480 321 1654">  </td> <td data-bbox="321 1480 1469 1654"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="240 1654 321 1829">  </td> <td data-bbox="321 1654 1469 1829"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>				
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>				


	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Hardwires is the array of references to hardwires on the virtual device</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 848 1468 1367"> <tr> <td data-bbox="240 848 321 1016">  </td> <td data-bbox="321 848 1468 1016"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1016 321 1184">  </td> <td data-bbox="321 1016 1468 1184"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1184 321 1367">  </td> <td data-bbox="321 1184 1468 1367"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseVirtualDevice Get Channels

Returns an array of channels on the virtual device.

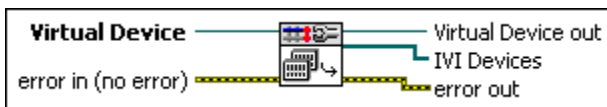








	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Channels is the array of references to the channels on the virtual device.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>

	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

niseVirtualDevice Get IVI Devices

Returns an array of IVI devices on the virtual device.

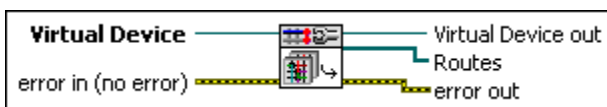


	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>	
<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>		
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>	
		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>	
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>	















	IVI Devices is the array of references to the IVI devices on the virtual device.	
error out contains error information. This output provides standard error out functionality .		
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.	
		code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.	

niseVirtualDevice Get Routes

Returns an array of routes on the virtual device.

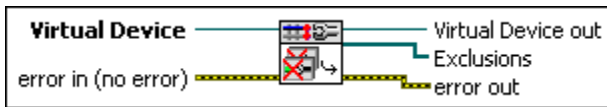


	Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.	
error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .		
		status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to




	<p>indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Routes is the array of references to the routes on the virtual device.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1150 1468 1671"> <tr> <td data-bbox="240 1150 321 1325">  </td> <td data-bbox="321 1150 1468 1325"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1325 321 1499">  </td> <td data-bbox="321 1325 1468 1499"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1499 321 1671">  </td> <td data-bbox="321 1499 1468 1671"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseVirtualDevice Get Exclusions

Returns an array of exclusions on the virtual device.

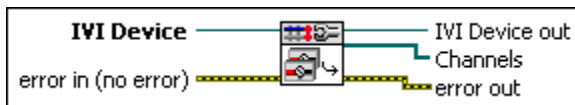





	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Exclusions is the array of references to the exclusions on the virtual device.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>














	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niselviDevice Get Channels

Returns an array of channels on the IVI device.

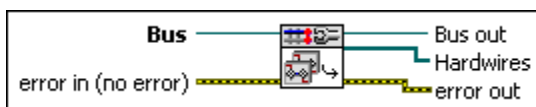


	<p>IVI Device is the reference to the IVI device. Call the niseVirtualDevice Get IVI Device VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>


	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>IVI Device out is the reference to the IVI device. Call the niseVirtualDevice Get IVI Device VI to obtain the reference.</p>						
	<p>Channels is the array of references to the channels on the IVI device.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 848 1468 1367"> <tr> <td data-bbox="240 848 321 1016">  </td> <td data-bbox="321 848 1468 1016"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1016 321 1184">  </td> <td data-bbox="321 1016 1468 1184"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1184 321 1367">  </td> <td data-bbox="321 1184 1468 1367"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseBus Get Hardwires

Returns an array of hardwires on the bus.

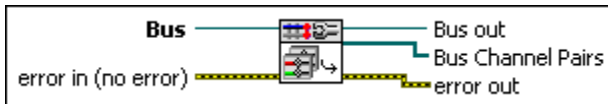








	Bus is the reference to the bus. Call the niseVirtualDevice Get Bus VI to obtain the reference.
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .
	status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
	code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
	Bus out is the reference to the bus. Call the niseVirtualDevice Get Bus VI to obtain the reference.
	Hardwires is the array of references to hardwires on the bus.
	error out contains error information. This output provides standard error out functionality .
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.

	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

niseBus Get Channel Pairs

Returns an array of bus channel pairs on the bus.

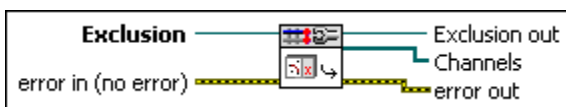


	<p>Bus is the reference to the bus. Call the niseVirtualDevice Get Bus VI to obtain the reference.</p>	
<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>		
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>	
		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>	
	<p>Bus out is the reference to the bus. Call the niseVirtualDevice Get Bus VI to obtain the reference.</p>	











	Bus Channel Pairs is the array of the references to the channel pairs on the bus.	
error out contains error information. This output provides standard error out functionality .		
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.	
		code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.	

niseExclusion Get Mutual Channels

Returns an array of excluded channels from the Mutual exclusion.

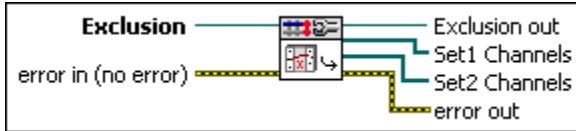


	Exclusion is the reference to the exclusion. Call the niseVirtualDevice Get Exclusion VI to obtain the reference.	
error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .		
		status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to






	<p>indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> <p> code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> <p> source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Exclusion out is the reference to the exclusion. Call the niseVirtualDevice Get Exclusion VI to obtain the reference.</p>
	<p>Channels is the array of references to the excluded channels.</p>
  	<p>error out contains error information. This output provides standard error out functionality.</p> <p> status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> <p> code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> <p> source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseExclusion Get Set Channels

Returns the arrays of excluded channels from the Set1 and Set2 collection of the set exclusion.





	<p>Exclusion is the reference to the exclusion. Call the niseVirtualDevice Get Exclusion VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Exclusion out is the reference to the exclusion. Call the niseVirtualDevice Get Exclusion VI to obtain the reference.</p>
	<p>Set1 Channels is the array of channel references from the Set1 collection of excluded channels.</p>
















	Set2 Channels is the array of channel references from the Set2 collection of excluded channels.	
error out contains error information. This output provides standard error out functionality .		
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.	
		code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.	

niseRoute Get Channels

Returns an array of constituent channels of the route.



	Route is the reference to the created route. Call the niseVirtualDevice Get Route VI to obtain the reference.
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .




	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Route out is the reference to the route. Call the niseVirtualDevice Get Route VI to obtain the reference.</p>						
	<p>Channels is the array of references to the constituent channels of the route.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1192 1468 1713"> <tr> <td data-bbox="240 1192 321 1367">  </td> <td data-bbox="321 1192 1468 1367"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1367 321 1541">  </td> <td data-bbox="321 1367 1468 1541"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1541 321 1713">  </td> <td data-bbox="321 1541 1468 1713"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseHardwire Get Channels

Returns an array of channels on the hardwire.

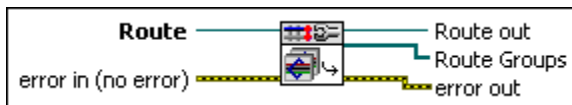










	<p>Hardwire is the reference to the hardwire. Call the niseVirtualDevice Get Hardwire VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Hardwire out is the reference to the hardwire. Call the niseVirtualDevice Get Hardwire VI to obtain the reference.</p>
	<p>Channels is the array of references to the channels on the hardwire.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>







	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseRoute Get Route Groups

Returns an array of parent route groups for the route.

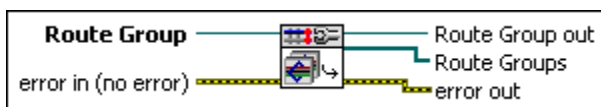


	<p>Route is the reference to the route. Call the niseVirtualDevice Get Route VI to obtain the reference.</p>				
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1480 1468 1829"> <tr> <td data-bbox="240 1480 321 1654">  </td> <td data-bbox="321 1480 1468 1654"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="240 1654 321 1829">  </td> <td data-bbox="321 1654 1468 1829"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>				
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>				


	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Route out is the reference to the route. Call the niseVirtualDevice Get Route VI to obtain the reference.</p>
	<p>Route Groups is the array of references to the parent route groups for the route.</p>
<p>error out contains error information. This output provides standard error out functionality.</p>	
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseRouteGroup Get Route Groups

Returns an array of the child route groups associated with the route group.

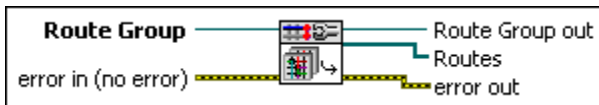






	<p>Route Group is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Route Group out is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.</p>
	<p>Route Groups is the array of references to the child route groups associated with the route group.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If</p>






	<p>status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseRouteGroup Get Routes

Returns all routes associated with the route group.



	<p>Route Group is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

	Route Group out is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.
	Routes is the array of references to the routes associated with the route group.
	error out contains error information. This output provides standard error out functionality .
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

niseCfg Close Object VI

Closes an object in the configuration.






Note All objects that you retrieve from the API must be closed using this VI when they are no longer needed. Failure to close an object results in a memory leak.






Reference specifies the reference to a NI Switch Executive object (or an array of references to NI Switch Executive objects) to release. When you are done with the NI Switch Executive

object, you must close the reference.

error in (no error) describes error conditions that occur before this node runs. This input provides standard [error in functionality](#).

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

error out contains error information. This output provides standard [error out functionality](#).

	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

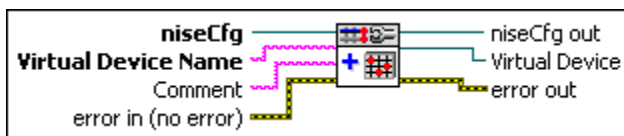
niseCfg Add Object VI

Adds an object to the collection.








niseCfg Add Virtual Device

Creates a new NI Switch Executive virtual device configuration.

The newly created virtual device is empty. Use the [niseVirtualDevice Add IVI Device VI](#) to add devices/channels to the configuration.



	niseCfg is the NI Switch Executive configuration session handle. Call the niseCfg Open Configuration VI to obtain the session handle.
	<p>Virtual Device Name specifies the name you want to assign to the newly created virtual device.</p> <p>The name must be unique—no other NI Switch Executive virtual device can have the same name. Refer to the Naming Conventions for a comprehensive list of NI Switch Executive naming rules.</p>
	Comment specifies the comment for the virtual device.
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .
	status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.

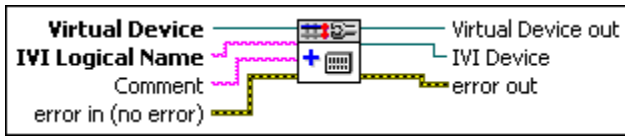
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>niseCfg out is the NI Switch Executive configuration session handle. Call the niseCfg Open Configuration VI to obtain the session handle.</p>
	<p>Virtual Device is the reference to the virtual device.</p>
  	<p>error out contains error information. This output provides standard error out functionality.</p> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Add IVI Device





Adds an IVI device to the virtual device configuration.

When NI Switch Executive adds an IVI device to a virtual device configuration, it queries

that device for its channels and adds those channels to the configuration as well.

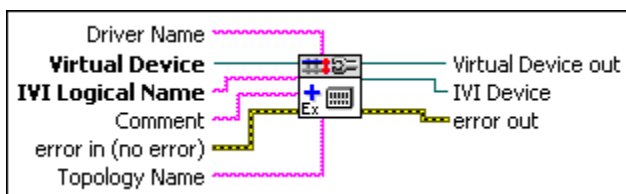



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>	
	<p>IVI Logical Name specifies the IVI Logical Name for the switch device you want to use in the configuration.</p>	
<p> Note IVI logical names are case sensitive.</p>		
	<p>Comment specifies the comment for the IVI device.</p>	
<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>		
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>	
		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>	
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI</p>	

	to obtain the reference.
	IVI Device is the reference to the created IVI device.
	error out contains error information. This output provides standard error out functionality .
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

niseVirtualDevice Add IVI Device Ex

Adds an IVI device to the virtual device configuration given a specific driver name and topology name.



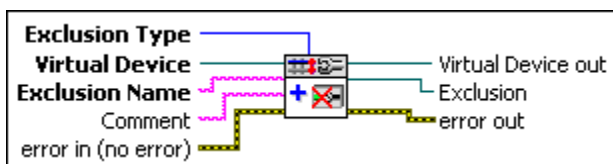
	Driver Name specifies the driver name for the IVI device. Valid values are "NI-DAQmx" or "IVI".
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------

	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>	
	<p>IVI Logical Name specifies the IVI Logical Name for the switch device you want to use in the configuration.</p>	
	<p>Note IVI logical names are case sensitive.</p>	
	<p>Comment specifies the comment for the IVI device.</p>	
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>	
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>	
		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>	
	<p>Topology Name specifies the topology name for the IVI device.</p>	
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>	

	IVI Device is the reference to the created IVI device.
error out contains error information. This output provides standard error out functionality .	
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.	
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.



niseVirtualDevice Add Exclusion

Creates a new exclusion and adds it to the virtual device.



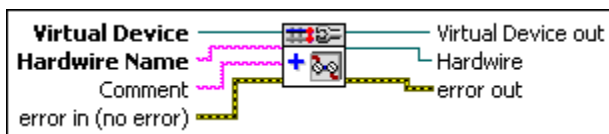
	Exclusion Type specifies the type of the exclusion. Refer to the Type parameter for a list of valid parameter values.
	Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.





	<p>Exclusion Name specifies the name for the exclusion you want to create.</p> <p>The name must be unique—no other exclusion can have the same name. Refer to the Naming Conventions for a comprehensive list of NI Switch Executive naming rules.</p>						
	<p>Comment specifies the comment for the exclusion.</p>						
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 688 1469 1207"> <tr> <td data-bbox="251 762 316 798"></td> <td data-bbox="337 741 1469 814"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="251 940 316 976"></td> <td data-bbox="337 919 1469 993"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="251 1119 316 1155"></td> <td data-bbox="337 1098 1469 1171"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Exclusion is the reference to the created exclusion.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1701 1469 1837"> <tr> <td data-bbox="251 1759 316 1795"></td> <td data-bbox="337 1759 1469 1791"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or</p>				
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or</p>						
















	that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

niseVirtualDevice Add Hardware

Creates a new hardware and adds it to the virtual device.

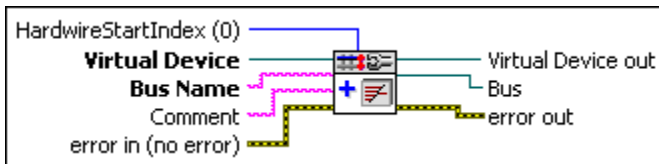










	Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.
	Hardware Name specifies the name for the hardware you want to create. The name must be unique—no other hardware can have the same name. Refer to the Naming Conventions for a comprehensive list of NI Switch Executive naming rules.
	Comment specifies the comment for the hardware.
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .















	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Hardwire is the reference to the created hardwire.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1192 1468 1713"> <tr> <td data-bbox="240 1192 321 1367">  </td> <td data-bbox="321 1192 1468 1367"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1367 321 1541">  </td> <td data-bbox="321 1367 1468 1541"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1541 321 1713">  </td> <td data-bbox="321 1541 1468 1713"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseVirtualDevice Add Bus

Creates a new bus and adds it to the virtual device.



	<p>HardwireStartIndex specifies the number that defines the names for the corresponding hardwires.</p> <p>When you create a bus, NI Switch Executive creates a corresponding set of hardwires. The names of these hardwires are defined by sequentially appending numbers to the bus name. The hardwire start index specifies the number appended to the bus name for the first hardwire in the corresponding set of hardwires.</p>		
	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>		
	<p>Bus Name specifies the name for the bus you want to create.</p> <p>The name must be unique—no other bus can have the same name. Refer to the Naming Conventions for a comprehensive list of NI Switch Executive naming rules.</p>		
	<p>Comment specifies the comment for the bus.</p>		
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1612 1468 1787"> <tr> <td data-bbox="250 1619 326 1780">  </td> <td data-bbox="326 1619 1468 1780"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		

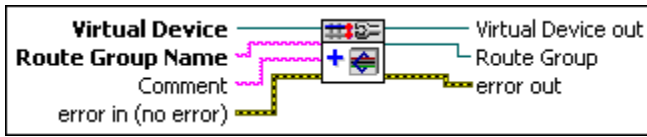
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Bus is the reference to the created bus.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1024 1468 1541"> <tr> <td data-bbox="240 1024 321 1192">  </td> <td data-bbox="321 1024 1468 1192"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1192 321 1367">  </td> <td data-bbox="321 1192 1468 1367"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1367 321 1541">  </td> <td data-bbox="321 1367 1468 1541"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseVirtualDevice Add Route Group

Creates a new route group and adds it to the virtual device.

Refer to [Additional Considerations](#) for more information about programmatic route

creation.



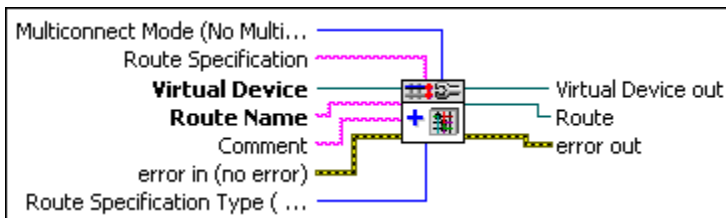
	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Route Group Name specifies the name for the route group you want to create.</p> <p>The name must be unique—no other route or a route group can have the same name. Refer to the Naming Conventions for a comprehensive list of NI Switch Executive naming rules.</p>						
	<p>Comment specifies the comment for the route group.</p>						
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1121 1468 1640"> <tr> <td data-bbox="250 1192 315 1226"></td> <td data-bbox="334 1171 1430 1245"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="250 1373 315 1407"></td> <td data-bbox="334 1346 1430 1419"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="250 1541 315 1575"></td> <td data-bbox="334 1514 1430 1587"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						

▶	Route Group is the reference to the created route group.	
error out contains error information. This output provides standard error out functionality .		
▶TF	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.	
▶I32	▶I32	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
▶abc	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.	

Virtual Device Add Route

Creates a new route and adds it to the virtual device.

See [Additional Considerations](#) for more information about programmatic route creation.



▶I32	Multiconnect Mode (No Multiconnect) specifies the connection mode for the route. Refer to the Multiconnect Mode property for a list of valid parameter values.
------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Route Specification specifies the route specification string you want to assign to the route.</p> <p>Refer to the Route Specification String examples for a comprehensive list of the naming rules that govern the route specification strings in NI Switch Executive. To compare and contrast route creation with the configuration API and the MAX UI, refer to the Route Specification property. The following actions do not cause an error in the configuration API:</p> <ul style="list-style-type: none"> • Route creation that uses channels with different wire modes • Use of valid syntax, without reserved for routing checks of the intermediate channels or checks of exclusion violation conflicts <p>To verify the configuration, open the configuration in MAX, load all routes and route groups in the Routes/Groups tab, and run the validator and/or the test panel.</p>				
	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>				
	<p>Route Name specifies the name for the route you want to create.</p> <p>The name must be unique—no other route or a route group can have the same name. Refer to the Naming Conventions for a comprehensive list of NI Switch Executive naming rules.</p>				
	<p>Comment specifies the comment for the route.</p>				
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1459 1464 1801"> <tr> <td data-bbox="251 1522 316 1564"></td> <td data-bbox="332 1501 1453 1627"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="251 1701 316 1743"></td> <td data-bbox="332 1680 1453 1764"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>				
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>				



source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.

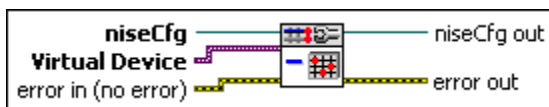
niseCfg Remove Object VI

Removes an object.

After NI Switch Executive removes an object, it does **not** need to be released using the [niseCfg Close Object VI](#).

niseCfg Remove Virtual Device

Deletes a virtual device from the configuration.



niseCfg is the NI Switch Executive configuration session handle. Call the [niseCfg Open Configuration VI](#) to obtain the session handle.




Virtual Device specifies the virtual device to remove.

Specify the virtual device using any of the following data types:






- The reference to the virtual device, which you can obtain with the [niseCfg Get Virtual Device VI](#)
- A one-based index to the collection of virtual devices in your NI Switch Executive configuration
- The name of the virtual device configuration

error in (no error) describes error conditions that occur before this node runs. This input provides standard [error in functionality](#).

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

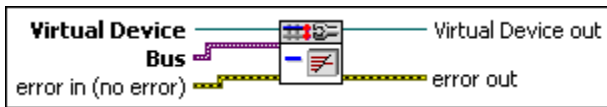
niseCfg out is the NI Switch Executive configuration session handle. Call the [niseCfg Open Configuration VI](#) to obtain the session handle.

error out contains error information. This output provides standard [error out functionality](#).

	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Remove Bus




Deletes a bus from the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Bus specifies the bus to remove from the virtual device.</p> <p>Specify the bus using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the bus, which you can obtain with the niseVirtualDevice Get Bus VI • A one-based index to the collection of buses on a virtual device • The bus name
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI</p>

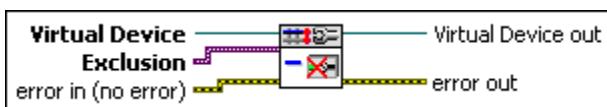
to obtain the reference.



error out contains error information. This output provides standard [error out functionality](#).

	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Remove Exclusion




Deletes an exclusion from the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Exclusion specifies the exclusion to remove from the virtual device.</p> <p>Specify the exclusion using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the exclusion, which you can obtain with the niseVirtualDevice Get Exclusion VI • A one-based index to the collection of exclusions on a virtual device




- The exclusion name

error in (no error) describes error conditions that occur before this node runs. This input provides standard [error in functionality](#).

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

Virtual Device out is the reference to the virtual device. Call the [niseCfg Get Virtual Device VI](#) to obtain the reference.

error out contains error information. This output provides standard [error out functionality](#).

	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Remove Hardware

Deletes a hardware from the virtual device.

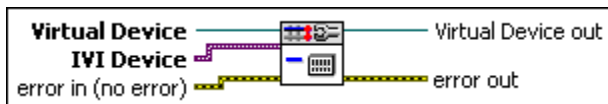


	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>Hardware specifies the hardware to remove from the virtual device.</p> <p>Specify the hardware using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the hardware, which you can obtain with the niseVirtualDevice Get Hardware VI • A one-based index to the collection of hardwires on a virtual device • The hardware name 						
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1209 1468 1728"> <tr> <td data-bbox="250 1283 315 1320"></td> <td data-bbox="334 1262 1430 1335"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="250 1461 315 1499"></td> <td data-bbox="334 1440 1414 1514"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="250 1629 315 1667"></td> <td data-bbox="334 1608 1451 1682"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						


	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Remove IVI Device

Deletes an IVI device from the virtual device.



	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>IVI Device specifies the IVI device to remove from the virtual device.</p> <p>Specify the IVI device using any of the following data types:</p> <ul style="list-style-type: none"> The reference to the IVI device, which you can obtain with the niseVirtualDevice Get IVI Device VI

	<ul style="list-style-type: none"> • A one-based index to the collection of IVI devices on a virtual device • The IVI device name 						
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 480 1468 999"> <tr> <td data-bbox="245 487 324 653">  </td> <td data-bbox="324 487 1463 653"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="245 653 324 827">  </td> <td data-bbox="324 653 1463 827"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="245 827 324 999">  </td> <td data-bbox="324 827 1463 999"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1360 1468 1839"> <tr> <td data-bbox="245 1367 324 1533">  </td> <td data-bbox="324 1367 1463 1533"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="245 1533 324 1707">  </td> <td data-bbox="324 1533 1463 1707"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="245 1707 324 1839">  </td> <td data-bbox="324 1707 1463 1839"> <p>source describes the origin of the error or warning and is, in most cases, the name of</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of</p>						







the VI that produced the error or warning.

niseVirtualDevice Remove Route Group

Deletes a route group from the virtual device.

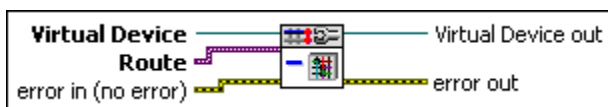


	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>				
	<p>Route Group specifies the route group to remove from the virtual device.</p> <p>Specify the route group using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the route group, which you can obtain with the niseVirtualDevice Get Route Group VI • A one-based index to the collection of route groups on a virtual device • The route group name 				
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1499 1468 1801"> <tr> <td data-bbox="250 1570 315 1600"></td> <td data-bbox="334 1556 1430 1625"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="250 1726 315 1755"></td> <td data-bbox="334 1724 1430 1759"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>				
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-</p>				




	<p>zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseVirtualDevice Remove Route

Deletes a route from the virtual device.

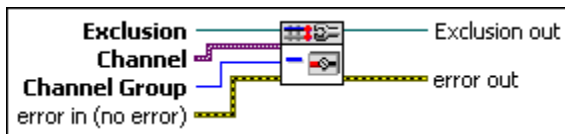





	<p>Virtual Device is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>Route specifies the route to remove from the virtual device.</p> <p>Specify the route using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the route, which you can obtain with the niseVirtualDevice Get Route VI • A one-based index to the collection of routes on a virtual device • The route name
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Virtual Device out is the reference to the virtual device. Call the niseCfg Get Virtual Device VI to obtain the reference.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>

	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseExclusion Remove Channel




Removes a channel from the specified set of excluded channels.



	<p>Exclusion is the reference to the exclusion. Call the niseVirtualDevice Get Exclusion VI to obtain the reference.</p>
	<p>Channel specifies the channel to remove from the exclusion collection.</p> <p>Specify the channel using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the channel, which you can obtain with the niseVirtual Device Get Channel VI • A one-based index to the collection of channels in the exclusion set • The channel name
	<p>Channel Group specifies the type of the collection of channels in the exclusion.</p>



If the exclusion type is 'Mutual Exclusion', you must specify 'mutual channels' as the type of the channel group. If the exclusion type is 'Set Exclusion', you must specify either Set1 or Set2 channels as the collection from which to remove the channel.


error in (no error) describes error conditions that occur before this node runs. This input provides standard [error in functionality](#).

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>

Exclusion out is the reference to the exclusion. Call the [niseVirtualDevice Get Exclusion VI](#) to obtain the reference.

error out contains error information. This output provides standard [error out functionality](#).





	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
















	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

niseHardwire Remove Channel

Removes the channel from the hardware.

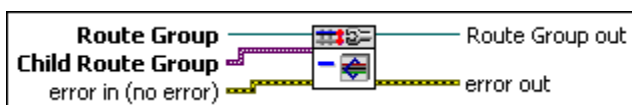


	<p>Hardware is the reference to the hardware. Call the niseVirtualDevice Get Hardware VI to obtain the reference.</p>
	<p>Channel specifies the channel to remove from the exclusion collection.</p> <p>Specify the channel using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the channel, which you can obtain with the niseVirtual Device Get Channel VI • A one-based index to the collection of channels in the exclusion set • The channel name
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>




	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Hardwire out is the reference to the hardwire. Call the niseVirtualDevice Get Hardwire VI to obtain the reference.</p>						
  	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 890 1468 1409"> <tr> <td data-bbox="240 890 321 1058">  </td> <td data-bbox="321 890 1468 1058"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1058 321 1234">  </td> <td data-bbox="321 1058 1468 1234"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1234 321 1409">  </td> <td data-bbox="321 1234 1468 1409"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseRouteGroup Remove Route Group

Disassociates the child route group from the route group.

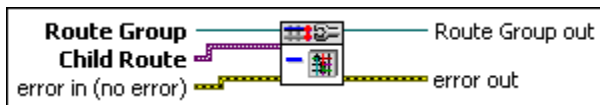





	<p>Route Group is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.</p>						
	<p>Child Route Group specifies the route group to remove from the collection of child route groups.</p> <p>Specify the route group using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the route group, which you can obtain with the niseVirtualDevice Get Route Group VI • A one-based index to the route group in the collection of child route groups • The route group name 						
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="235 924 1469 1438"> <tr> <td data-bbox="251 997 316 1039"></td> <td data-bbox="332 976 1437 1060"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="251 1165 316 1207"></td> <td data-bbox="332 1144 1421 1228"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="251 1344 316 1386"></td> <td data-bbox="332 1323 1453 1407"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Route Group out is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p>						

	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

niseRouteGroup Remove Route

Disassociates the child route from the route group.



	Route Group is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.
	<p>Child Route specifies the route to remove from the child route groups collection.</p> <p>Specify the route using any of the following data types:</p> <ul style="list-style-type: none"> • The reference to the route, which you can obtain with the niseVirtualDevice Get Route VI • A one-based index to the route group in the collection of child routes • The route name
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .




	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>Route Group out is the reference to the route group. Call the niseVirtualDevice Get Route Group VI to obtain the reference.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1062 1468 1577"> <tr> <td data-bbox="240 1062 321 1236">  </td> <td data-bbox="321 1062 1468 1236"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1236 321 1411">  </td> <td data-bbox="321 1236 1468 1411"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1411 321 1577">  </td> <td data-bbox="321 1411 1468 1577"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseBus Remove Channel Pair

Removes a channel pair from a bus.

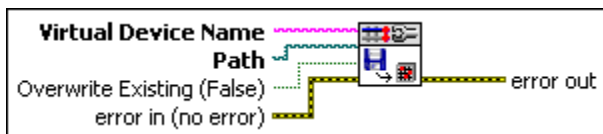





	Bus is the reference to the bus. Call the niseVirtualDevice Get Bus VI to obtain the reference.
	<p>Bus Channel Pair specifies the reference to the channel pair to remove from the bus.</p> <p>You can pass a one-based index to the collection of bus channel pairs as this parameter. Alternatively, you can pass the bus channel pair reference from the array of references you obtain with the niseBus Get Channel Pairs VI. You cannot refer to a bus channel pair by its name.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>Bus out is the reference to the bus. Call the niseVirtualDevice Get Bus VI to obtain the reference.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>



















	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

niseCfg Import VI

Imports a virtual device configuration from an NI Switch Executive configuration file.

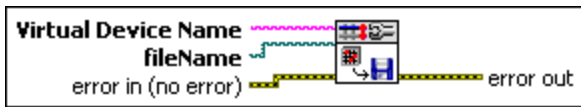


	Virtual Device Name specifies the name you want to assign to the imported virtual device configuration.
	Path specifies the full path to the filename that contains the exported NI Switch Executive virtual device configuration.
	Overwrite Existing (False) specifies whether to overwrite the existing configuration. Pass True to replace any existing virtual device configurations with the same name. Pass False to return an error if a virtual device configuration with the same name already exists.


	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 306 1468 827"> <tr> <td data-bbox="240 306 326 478"></td> <td data-bbox="326 306 1468 478">status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</td> </tr> <tr> <td data-bbox="240 478 326 651"></td> <td data-bbox="326 478 1468 651">code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</td> </tr> <tr> <td data-bbox="240 651 326 827"></td> <td data-bbox="326 651 1468 827">source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</td> </tr> </table>		status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.		code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.		source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
	status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.						
	code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.						
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 1012 1468 1533"> <tr> <td data-bbox="240 1012 326 1184"></td> <td data-bbox="326 1012 1468 1184">status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</td> </tr> <tr> <td data-bbox="240 1184 326 1356"></td> <td data-bbox="326 1184 1468 1356">code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</td> </tr> <tr> <td data-bbox="240 1356 326 1533"></td> <td data-bbox="326 1356 1468 1533">source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</td> </tr> </table>		status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.		code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.		source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.						
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.						
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.						

niseCfg Export VI

Exports the virtual device configuration to an NI Switch Executive Export File.

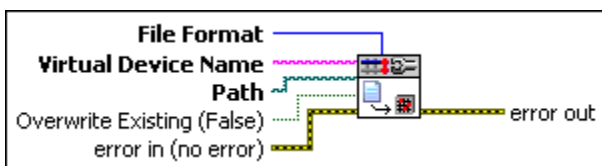







	Virtual Device Name specifies the name of the virtual device.
	fileName specifies the full path to the file to which to export the NI Switch Executive virtual device configuration.
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .
	status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
	code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
	error out contains error information. This output provides standard error out functionality .
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.














	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

niseCfg Import Specific VI

Imports a virtual device configuration from a file.

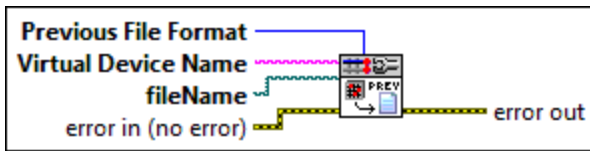


	<p>File Format specifies the file format that contains the NI Switch Executive virtual device configuration.</p>
	<p>Virtual Device Name specifies the name you want to assign to the imported virtual device configuration.</p>
	<p>Path specifies the full path to the filename that contains the exported NI Switch Executive virtual device configuration.</p>
	<p>Overwrite Existing (False) specifies whether to overwrite the existing configuration. Pass True to replace any existing virtual device configurations with the same name. Pass False to return an error if a virtual device configuration with the same name already exists.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>

	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>error out contains error information. This output provides standard error out functionality.</p> <table border="1" data-bbox="240 890 1468 1409"> <tr> <td data-bbox="240 890 321 1064">  </td> <td data-bbox="321 890 1468 1064"> <p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p> </td> </tr> <tr> <td data-bbox="240 1064 321 1239">  </td> <td data-bbox="321 1064 1468 1239"> <p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="240 1239 321 1409">  </td> <td data-bbox="321 1239 1468 1409"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>		<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>						
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>						

niseCfg Export Previous VI

Exports the virtual device configuration to a file format coinciding with a previous version of NI Switch Executive.



Previous File Format specifies the previous NI Switch Executive version file format to which to export the NI Switch Executive virtual device configuration.

The available file formats are as follows:

<p>NI Switch Executive 2.0 Export File (1)</p>	<p>Exports the NI Switch Executive configuration to an NI Switch Executive 2.0 file (XML).</p>
<p>NI Switch Executive 2.1 Export File (2)</p>	<p>Exports the NI Switch Executive configuration to an NI Switch Executive 2.1 file (XML).</p>
<p>NI Switch Executive 3.0 Export File (3)</p>	<p>Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 file (XML).</p>
<p>NI Switch Executive 3.0 Excel 97-2003 Workbook (4)</p>	<p>Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 Excel 97-2003 file.</p>
<p>NI Switch Executive 3.0 Excel 2007 Workbook (5)</p>	<p>Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 Excel 2007 file.</p>
<p>NI Switch Executive 3.0 Tab-delimited File (6)</p>	<p>Exports the NI Switch Executive configuration to an NI Switch Executive 3.0 tab delimited file.</p>

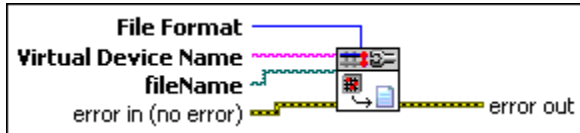


Virtual Device Name specifies the name of the virtual device.




	<p>fileName specifies the full path to the file to which to export the NI Switch Executive virtual device configuration.</p>
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>error out contains error information. This output provides standard error out functionality.</p>
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niseCfg Export Specific VI

Exports the virtual device configuration to a specified file format.



	File Format specifies the file format to which to export the NI Switch Executive virtual device configuration.	
	Virtual Device Name specifies the name of the virtual device.	
	fileName specifies the full path to the file to which to export the NI Switch Executive virtual device configuration.	
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .	
		status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
		code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
		source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
	error out contains error information. This output provides standard error out functionality .	

	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
	code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

Property Nodes

Owning Palette: [Configuration](#)

Use the NI Switch Executive property nodes to access properties of individual objects of the Switch Executive Virtual Device.

Low Level

Owning Palette: [NI Switch Executive VIs](#)


Use the VIs on the IVI and NI-SWITCH subpalettes to access IVI sessions to instrument drivers of switch devices that are a part of a session to the Switch Executive Virtual Device.

Subpalette	Description
IVI	Use the VIs on the IVI subpalette to access IVI sessions.
NI-SWITCH	Use the VIs on the NI-SWITCH subpalette to access NI-SWITCH sessions.

IVI


Owning Palette: [Low Level](#)

Use the VIs on the IVI subpalette to access IVI sessions.

Palette Object	Description
niSE Get IVI Device Session VI	<p>Retrieves an IVI instrument session for an IVI switching device that is being managed by NI Switch Executive. The retrieved session handle can be used to access instrument specific functionality through the instrument driver.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>Note Use caution when using the session handle. Calling functions on an instrument driver can invalidate the NI Switch Executive configuration and cache. You should not use the retrieved session handle to make or break connections or modify the configuration channels, as doing so can cause undefined, and potentially unwanted, behavior.</p> </div> <p>When you finish using the IVI device session, call the niSE Release IVI Device Session VI.</p>
niSE Release IVI Device Session VI	<p>Releases an IVI instrument session retrieved by a previous call to the niSE Get IVI Device Session VI.</p>

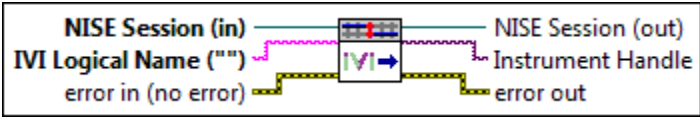
niSE Get IVI Device Session VI

Retrieves an IVI instrument session for an IVI switching device that is being managed by NI Switch Executive. The retrieved session handle can be used to access instrument specific functionality through the instrument driver.

	<p>Note Use caution when using the session handle. Calling functions on an instrument driver can invalidate the NI Switch Executive configuration and cache. You should not use the retrieved session handle to make or break connections or modify the configuration channels,</p>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

as doing so can cause undefined, and potentially unwanted, behavior.

When you finish using the IVI device session, call the [niSE Release IVI Device Session VI](#).

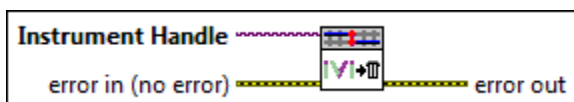


	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>						
	<p>IVI Logical Name ("") is the IVI logical name of the IVI device for which to retrieve an IVI session.</p>						
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p> <table border="1" data-bbox="240 1010 1468 1530"> <tr> <td data-bbox="250 1010 326 1184"> </td> <td data-bbox="326 1010 1468 1184"> <p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p> </td> </tr> <tr> <td data-bbox="250 1184 326 1358"> </td> <td data-bbox="326 1184 1468 1358"> <p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p> </td> </tr> <tr> <td data-bbox="250 1358 326 1530"> </td> <td data-bbox="326 1358 1468 1530"> <p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p> </td> </tr> </table>		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>						
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>						
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>						
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>						







	Instrument Handle is the IVI instrument handle of the specified IVI device.	
	error out contains error information. This output provides standard error out functionality .	
	status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.	
		code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
	source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.	

niSE Release IVI Device Session VI

Releases an IVI instrument session retrieved by a previous call to the [niSE Get IVI Device Session VI](#).



	Instrument Handle is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to niSE Open Session VI .
	error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality .


	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
<p>error out contains error information. This output provides standard error out functionality.</p>	
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

NI-SWITCH

Owning Palette: [Low Level](#)


Use the VIs on the NI-SWITCH subpalette to access NI-SWITCH sessions.

Palette	Description
---------	-------------

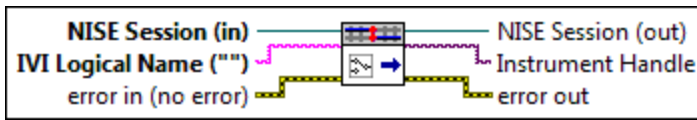
Object	
niSE Get NI-SWITCH Session VI	<p>Retrieves an NI-SWITCH instrument session for an NI-SWITCH switching device that is being managed by NI Switch Executive. The retrieved session handle can be used to access instrument specific functionality through the instrument driver.</p> <div data-bbox="280 409 1466 630" style="border: 1px solid black; padding: 5px;">  <p>Note Use caution when using the session handle. Calling functions on an instrument driver can invalidate the NI Switch Executive configuration and cache. You should not use the retrieved session handle to make or break connections or modify the configuration channels, as doing so can cause undefined, and potentially unwanted, behavior.</p> </div> <p>When you finish using the NI-SWITCH device session, call the niSE Release NI-SWITCH Session VI.</p>
niSE Release NI-SWITCH Session VI	<p>Releases an NI-SWITCH instrument session retrieved by a previous call to the niSE Get NI-SWITCH Session VI.</p>

niSE Get NI-SWITCH Session VI




Retrieves an NI-SWITCH instrument session for an NI-SWITCH switching device that is being managed by NI Switch Executive. The retrieved session handle can be used to access instrument specific functionality through the instrument driver.

	<p>Note Use caution when using the session handle. Calling functions on an instrument driver can invalidate the NI Switch Executive configuration and cache. You should not use the retrieved session handle to make or break connections or modify the configuration channels, as doing so can cause undefined, and potentially unwanted, behavior.</p>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

When you finish using the NI-SWITCH device session, call the [niSE Release NI-SWITCH Session VI](#).





	<p>NISE Session (in) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI.</p>	
	<p>IVI Logical Name (\"'\") is the IVI logical name of the NI-SWITCH device for which to retrieve an NI-SWITCH session.</p>	
	<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>	
		<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>
		<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
		<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
	<p>NISE Session (out) is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to the niSE Open Session VI. This session handle is the copy of the session handle that was passed in and provides for easier wiring between NI Switch Executive VIs.</p>	
	<p>Instrument Handle is the IVI instrument handle of the specified IVI device.</p>	






<p>error out contains error information. This output provides standard error out functionality.</p>	
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

niSE Release NI-SWITCH Session VI

Releases an NI-SWITCH instrument session retrieved by a previous call to the [niSE Get NI-SWITCH Session VI](#).



	<p>Instrument Handle is the session referencing this NI Switch Executive virtual device. Session handles are created through a call to niSE Open Session VI.</p>
<p>error in (no error) describes error conditions that occur before this node runs. This input provides standard error in functionality.</p>	
	<p>status is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.</p>

	<p>code is the error or warning code. The default is 0. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.</p>
<p>error out contains error information. This output provides standard error out functionality.</p>	
	<p>status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.</p>
	<p>code is the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.</p>
	<p>source describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.</p>

Using the Standard Functionality for error in Parameters

Many LabVIEW nodes such as VIs contain **error in** parameters you can use to manage errors. These parameters typically provide the same, standard functionality. When a node exhibits different parameter functionality, the exceptions are documented in the reference material for that node. Standard **error in** behavior is as follows



Note Some nodes, such as error handling VIs, contain an **error in** parameter that does not provide standard error in functionality, but that contains an **error in** cluster that is standard.






error in describes error conditions that occur before this node runs. The default is no erro

r. If an error occurred before this node runs, the node passes the **error in** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**.

Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.


The **error in** cluster contains the following cluster elements:

	status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.
	code is the error or warning code. The default is 0. If status is TRUE, code is an error code. If status is FALSE, code is 0 or a warning code.
	source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.


Using the Standard Functionality for error out Parameters



Many LabVIEW nodes such as VIs contain an **error out** parameter you can use to manage errors. These parameters typically provide the same, standard functionality. When a node exhibits different parameter functionality, the exceptions are documented in the reference material for that node.

Standard **error out** functionality is as follows:

	error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the error out front panel indicator and select Explain Error from the shortcut menu for more information about the error.
-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------


error out contains the following cluster elements:

	status is TRUE (X) if an error occurred before this node ran or during the running of this node or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran or during the running of this node.
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	code is the error or warning code. If status is TRUE, code is an error code. If status is FALSE, code is 0 or a warning code.
	source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

NI Switch Executive C Function Reference

Use the NI Switch Executive functions to create an application using LabWindows™/CVI™ or Microsoft Visual C++.

	<p>Note All of the following functions return <code>niSE_Status</code>. A value of 0 indicates that the function was successful. Other values represent error conditions and are available from the Error Codes table and/or calls to niSE_GetError.</p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- [niSE_OpenSession](#)
- [niSE_CloseSession](#)
- [niSE_Connect](#)
- [niSE_ConnectAndDisconnect](#)
- [niSE_Disconnect](#)
- [niSE_DisconnectAll](#)
- [niSE_ExpandRouteSpec](#)
- [niSE_FindRoute](#)
- [niSE_GetAllConnections](#)
- [niSE_IsConnected](#)
- [niSE_IsDebounced](#)
- [niSE_WaitForDebounce](#)
- [niSE_GetIviDeviceSession](#)
- [niSE_GetError](#)
- [niSE_ClearError](#)

niSE_ClearError

Clears the last queried error from memory.

Function Prototype

```
NISEStatus __stdcall niSE_ClearError(  
    NISESession sessionHandle);
```

Parameter

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

niSE_CloseSession

Reduces the reference count of open sessions by one. If the reference count goes to 0, the function deallocates any memory resources the driver uses and closes any open IVI switch sessions.

Function Prototype

```
NISEStatus __stdcall niSE_CloseSession(  
    NISESession sessionHandle);
```

Parameter

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

niSE_CloseSession Details

After calling the `niSE_CloseSession` function, you should not use the NI Switch Executive virtual device again until you call [niSE_OpenSession](#).

niSE_Connect

Connects the routes specified by the connection specification. When connecting, it may allow for multiconnection based on the multiconnection mode.

Function Prototype

NISEStatus __stdcall niSE_Connect(<i>NISESession sessionHandle,</i>
	<i>NISEConstString connectSpec,</i>
	<i>NISEInt32 multiconnectMode,</i>
	<i>NISEBoolean waitForDebounce);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
connectSpec	NISEConstString	The string describing the connections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.
multiconnectMode	NISEInt32	This value sets the connection mode for the function. The mode may be one of the following: <ul style="list-style-type: none"> NISE_VAL_USE_DEFAULT_MODE (-1)—uses the mode selected as the default for the route in the NI Switch Executive virtual device configuration. If a mode has not been selected for the route in the NI Switch Executive virtual device, this parameter defaults to NISE_VAL_MULTICONNECT_ROUTES. NISE_VAL_NO_MULTICONNECT (0)—routes specified in the connection specification must be disconnected

		<p>before they can be reconnected. Calling <code>Connect</code> on a route that was connected using No Multiconnect mode results in an error condition.</p> <ul style="list-style-type: none"> • <code>NISE_VAL_MULTICONNECT_ROUTES (1)</code>—routes specified in the connection specification can be connected multiple times. The first call to <code>Connect</code> performs the physical hardware connection. Successive calls to <code>Connect</code> increase a connection reference count. Similarly, calls to <code>Disconnect</code> decrease the reference count. When it reaches 0, the hardware is physically disconnected. Multiconnecting routes applies to entire routes and not to route segments.
waitForDebounce	<code>NISEBoolean</code>	<p>Waits (if true) for switches to debounce between its connect and disconnect operations. If false, it immediately begins the second operation after completing the first. The order of connect and disconnect operation is set by the Operation Order input.</p>

niSE_Connect Details

In the event of an error, the call to `niSE_Connect` attempts to undo any connections made so that the system is left in the same state that it was in before the call was made. Some errors can be caught before manipulating hardware, although it is feasible that a hardware call could fail causing some connections to be momentarily closed and then reopened.

If the wait for debounce parameter is set, the function does not return until the switch system has debounced.

niSE_ConnectAndDisconnect

Connects routes and disconnects routes in a similar fashion to `niSE_Connect` and `niSE_Disconnect` except that the operations happen in the context of a single function call. This function is useful for switching from one state to another state.

This function will not wait for debounce. Call the `niSE_WaitForDebounce` function to wait for debounce.

Function Prototype

NISEStatus __stdcall niSE_ConnectAndDisconnect(<i>NISESession sessionHandle,</i>
	<i>NISEConstString connectSpec,</i>
	<i>NISEConstString disconnectSpec,</i>
	<i>NISEInt32 multiconnectMode,</i>
	<i>NISEInt32 operationOrder,</i>
	<i>NISEBoolean waitForDebounce);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
connectSpec	NISEConstString	The string describing the connections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.
disconnectSpec	NISEConstString	The string describing the disconnections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.
multiconnectMode	NISEInt32	This value sets the connection mode for the function. The mode may be one of the following: <ul style="list-style-type: none"> NISE_VAL_USE_DEFAULT_MODE (-1)—uses the mode selected as the default for the route in the NI Switch Executive virtual device configuration. If a mode has not been selected for the route in the

		<p>NI Switch Executive virtual device, this parameter defaults to NISE_VAL_MULTICONNECT_ROUTES.</p> <ul style="list-style-type: none"> • NISE_VAL_NO_MULTICONNECT (0)—routes specified in the connection specification must be disconnected before they can be reconnected. Calling <code>Connect</code> on a route that was connected using No Multiconnect mode results in an error condition. • NISE_VAL_MULTICONNECT_ROUTES (1)—routes specified in the connection specification can be connected multiple times. The first call to <code>Connect</code> performs the physical hardware connection. Successive calls to <code>Connect</code> increase a connection reference count. Similarly, calls to <code>Disconnect</code> decrease the reference count. When it reaches 0, the hardware is physically disconnected. This behavior is slightly different with SPDT relays. For more information, refer to Exclusions and SPDT Relays. Multiconnecting routes applies to entire routes and not to route segments.
<p>operationOrder</p>	<p>NISEInt32</p>	<p>Sets the order of the operation for the function. Defined values are Break Before Make and Break After Make.</p> <ul style="list-style-type: none"> • NISE_VAL_BREAK_BEFORE_MAKE (1)—The function disconnects the routes specified in the disconnect specification before connecting the routes specified in the connect specification. This is the typical mode of operation. <p>For NISE_VAL_BREAK_BEFORE_MAKE operation order, an asterisk (*) will disconnect all connections currently connected and connect what is in the connect specification. However, this action maintains as closed those connections currently connected that also exist in the disconnect specification. This behavior minimizes relay actuation and thus extends the life of the relays.</p> <ul style="list-style-type: none"> • NISE_VAL_BREAK_AFTER_MAKE (2)—The function connects the routes specified in the connection specification before connecting the routes specified in the disconnection specification. This mode of

		<p>operation is normally used when you are switching current and want to ensure that a load is always connected to your source. The order of operation is to connect first or disconnect first.</p> <p>For NISE_VAL_BREAK_AFTER_MAKE operation order, the VI connects what is in the connect specification and the asterisk (*) will disconnect all other connections. As in case 1, this action maintains as closed those connections currently connected that also exist in the connect specification. This behavior minimizes relay actuation and thus extends the life of the relays.</p>
waitForDebounce	NISEBoolean	<p>Waits (if true) for switches to debounce between its connect and disconnect operations. If false, it immediately begins the second operation after completing the first. The order of connect and disconnect operation is set by the Operation Order input.</p>

niSE_ConnectAndDisconnect Details

`niSE_ConnectAndDisconnect` manipulates the hardware connections and disconnections only when the routes are different between the connection and disconnection specifications. If any routes are common between the connection and disconnection specifications, NI Switch Executive determines whether the relays need to be switched. This functionality has the distinct advantage of increased throughput for shared connections, because hardware does not have to be involved and potentially increases relay lifetime by decreasing the number of times that the relay has to be switched.

In the event of an error, the call to `niSE_ConnectAndDisconnect` attempts to undo any connections made, but does not attempt to reconnect disconnections. Some errors can be caught before manipulating hardware, although it is feasible that a hardware call could fail causing some connections to be momentarily closed and then reopened.

niSE_Disconnect

Disconnects the routes specified in the Disconnection Specification.

This function will not wait for debounce. Call the `niSE_WaitForDebounce` function to wait for debounce.

Function Prototype

NISEStatus __stdcall niSE_Disconnect(<i>NISESession sessionHandle,</i>
	<i>NISEConstString disconnectSpec);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
disconnectSpec	NISEConstString	The string describing the disconnections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.

niSE_Disconnect Details

If any of the specified routes were originally connected in a multiconnected mode, the call to `niSE_Disconnect` reduces the reference count on the route by 1. If the reference count reaches 0, it is disconnected.

If a specified route does not exist, it is an error condition.

In the event of an error, the call to `niSE_Disconnect` continues to try to disconnect everything specified by the route specification string but reports the error on completion.

niSE_DisconnectAll

Disconnects all connections on every IVI switch device managed by the NISE session reference passed to this function.

This function will not wait for debounce. Call the `niSE_WaitForDebounce` function to wait for debounce.

Function Prototype

NISEStatus __stdcall niSE_DisconnectAll(<i>NISESession sessionHandle</i>);
------------------------------------------	--------------------------------------------

Parameter

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

niSE_DisconnectAll Details

`niSE_DisconnectAll` ignores all multiconnect modes. Calling `niSE_DisconnectAll` resets all of the switch states for the system.

niSE_ExpandRouteSpec

Expands a route spec string to yield more information about the routes and route groups within the spec.

Function Prototype

NISEStatus __stdcall niSE_ExpandRouteSpec(<i>NISESession sessionHandle</i> ,
	<i>NISEConstString routeSpec</i> ,
	<i>NISEInt32 expandAction</i> ,

	<i>NISEBuffer * expandedRouteSpec,</i>
	<i>NISEInt32 * expandedRouteSpecSize);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
routeSpec	String	The routes or route groups to be expanded. Refer to Route Specification Strings for more information.
expandAction	NISEExpandAction	The mode can be one of the following: <ul style="list-style-type: none"> Expand to Routes (0)—expands the route spec to routes. Converts route groups to their constituent routes. Expand to Paths (1)—expands the route spec to paths. Converts routes and route groups to their constituent square bracket route spec strings.

Output

Name	Type	Description
expandedRouteSpec	NISEBuffer*	The expanded routes and route groups. Refer to Route Specification Strings for more information.
expandedRouteSpecSize	NISEInt32	<p>The expandedRouteSpecSize is an NISEInt32 that is passed by reference into the function.</p> <p>As an input, it is the size of the route string buffer being passed. If the route string is larger than the string buffer being passed, only the portion of the route string that can fit in the string buffer is copied into it.</p> <p>On return from the function, routeSpecSize holds the size required to hold the entire route string. This size may be larger than the buffer size as the function always returns the size needed to hold the entire buffer.</p>

		You can pass NULL for this parameter if you are not interested in the return value for routeSpecSize and routeSpec.
--	--	---------------------------------------------------------------------------------------------------------------------

niSE_ExpandRouteSpec Details

Use this function in conjunction with `niSE_GetAllConnections` to return the details of the currently connected routes and route groups. Then select, display, or error check what routes, devices, paths, and channels are currently connected based on the route details.

niSE_FindRoute

Finds an existing or potential route between channel 1 and channel 2.

Function Prototype

NISEStatus __stdcall niSE_FindRoute(<i>NISESession sessionHandle,</i>
	<i>NISEConstString channel1,</i>
	<i>NISEConstString channel2,</i>
	<i>NISEBuffer* routeSpec,</i>
	<i>NISEInt32 * routeSpecSize,</i>
	<i>NISEInt32 * routeCapability);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
channel1	NISEConstString	The channel name of one of the endpoints of the route to find. The channel name must either be a channel alias name or a

		<p>name in the device/ivichannel syntax.</p> <p>Examples: MyChannel Switch1/R0</p>
channel2	NISEConstString	<p>The channel name of one of the endpoints of the route to find. The channel name must either be a channel alias name or a name in the device/ivichannel syntax.</p> <p>Examples: MyChannel Switch1/R0</p>

Input/Output

Name	Type	Description
routeSpecSize	NISEInt32	<p>The routeSpecSize is an NISEInt32 that is passed by reference into the function.</p> <p>As an input, it is the size of the route string buffer being passed. If the route string is larger than the string buffer being passed, only the portion of the route string that can fit in the string buffer is copied into it.</p> <p>On return from the function, routeSpecSize holds the size required to hold the entire route string. This size may be larger than the buffer size as the function always returns the size needed to hold the entire buffer.</p> <p>You can pass NULL for this parameter if you are not interested in the return value for routeSpecSize and routeSpec.</p>

Output

Name	Type	Description
routeSpec	NISEBuffer*	The fully specified route path surrounded by delimiting square brackets—if the route exists or is possible. Refer to Route Specification Strings for more information.
routeCapability	NISEInt32	The return value which expresses the capability of finding a valid

		route between Channel 1 and Channel 2. Refer to the routeCapability table for value descriptions.
--	--	-------------------------------------------------------------------------------------------------------------------

niSE_FindRoute Details

The returned route specification contains the route specification, and the route capability indicates whether the route existed, is possible, or is not possible for various reasons.

The route specification string returned from `niSE_FindRoute` can be passed to other NI Switch Executive API functions (such as `niSE_Connect`, `niSE_Disconnect`, and `niSE_ConnectAndDisconnect`) that use route specification strings.

To dynamically allocate space for the **routeSpec** buffer, call `niSE_FindRoute` twice. The first call should specify a **routeSpecSize** of 0 and then use the returned **routeSpecSize** to allocate the correct buffer size.

routeCapability

routeCapability may be any one of the following:

Value	Name	Description
1	NISE_VAL_PATH_AVAILABLE	A path between channel 1 and channel 2 is available. The route specification parameter returns a string describing the available path.
2	NISE_VAL_PATH_EXISTS	A path between channel 1 and channel 2 already exists. The route specification parameter returns a string describing the existing path.
3	NISE_VAL_PATH_UNSUPPORTED	There is no potential path between channel 1 and channel 2 given the current configuration.
4	NISE_VAL_RSRC_IN_USE	There is a potential path between channel 1 and channel 2, although a resource needed to complete the path is already in use.
5	NISE_VAL_EXCLUSION_CONFLICT	Channel 1 and channel 2 cannot be connected because their connection would result in an exclusion violation.

6	NISE_VAL_CHANNEL_NOT_AVAILABLE	One of the channels is not useable as an endpoint channel. Make sure that it is not marked as a reserved for routing.
7	NISE_VAL_CHANNELS_HARDWIRED	The two channels reside on the same hardware. An implicit path already exists.

niSE_GetAllConnections

Returns all connected routes and route groups.

Function Prototype

NISEStatus __stdcall niSE_GetAllConnections(<i>NISESession sessionHandle,</i>
	<i>NISEBuffer* routeSpec,</i>
	<i>NISEInt32 * routeSpecSize);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

Output

Name	Type	Description
routeSpec	NISEBuffer*	The currently connected routes and route groups. Refer to Route Specification Strings for more information.
routeSpecSize	NISEInt32	The routeSpecSize is an NISEInt32 that is passed by reference into the function. As an input, it is the size of the route string buffer being passed. If the route string is larger than the string buffer being passed, only the portion of the route string that can fit in the string buffer is copied into it.

		<p>On return from the function, routeSpecSize holds the size required to hold the entire route string. This size may be larger than the buffer size as the function always returns the size needed to hold the entire buffer.</p> <p>You can pass NULL for this parameter if you are not interested in the return value for routeSpecSize and routeSpec.</p>
--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

niSE_GetAllConnections Details

You can pass the routeSpec to other functions that take a routeSpec as input, such as niSE_Connect, niSE_Disconnect, and niSE_ExpandRouteSpec.

niSE_GetError

Queries for and returns the most recent error.

Function Prototype

NISEStatus __stdcall niSE_GetError(<i>NISESession sessionHandle,</i>
	<i>NISEStatus * errorNumber,</i>
	<i>NISEBuffer* errorDescription,</i>
	<i>NISEInt32 * errorDescriptionSize);</i>


Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

Input/Output

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

errorDescriptionSize	NISEInt32	<p>The errorDescriptionSize is an NISEInt32 that is passed by reference into the function.</p> <p>As an input, it is the size of the error description buffer being passed. If the error description is larger than the error description buffer being passed, only the portion of the error description that can fit in the error description buffer is copied into it.</p> <p>On return from the function, errorDescriptionSize holds the size required to hold the entire error description.</p> <div data-bbox="610 621 1466 762" style="border: 1px solid black; padding: 5px;">  <p>Note errorDescriptionSize may be larger than the buffer size as the function always returns the size needed to hold the entire buffer.</p> </div> <p>You can pass NULL for this parameter if you are not interested in the return value for errorDescriptionSize and errorDescription.</p>
-----------------------------	-----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Output

Name	Type	Description
errorNumber	NISEStatus	The error code.
errorDescription	NISEBuffer	Description of the error.

To dynamically allocate space for the **errorDescription** buffer, call `niSE_GetError` twice. The first call should specify a **errorDescriptionSize** of 0 and then use the returned **errorDescriptionSize** to allocate the correct buffer size.

niSE_GetIviDeviceSession

Retrieves an IVI instrument session for an IVI switching device that is being managed by NI Switch Executive. The retrieved session handle can be used to access instrument specific functionality through the instrument driver.

The retrieved handle should **not** be closed.

Function Prototype

NISEStatus __stdcall niSE_GetIviDeviceSession(<i>NISESession sessionHandle,</i>
	<i>NISEConstString iviLogicalName,</i>
	<i>ViSession * iviSessionHandle);</i>

Parameters


Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
iviLogicalName	NISEConstString	The IVI logical name of the IVI device for which to retrieve an IVI session.

Output

Name	Type	Description
iviSessionHandle	ViSession	The IVI instrument handle of the specified IVI device.

niSE_GetIviDeviceSession Details

	Note Use caution when using the session handle. Calling functions on an instrument driver can invalidate the NI Switch Executive configuration and cache. You should not use the retrieved session handle to make or break connections or modify the configuration channels, as doing so can cause undefined, and potentially unwanted, behavior.
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

niSE_IsConnected

Checks to see whether the specified route or route group is connected.

Function Prototype

NISEStatus __stdcall niSE_IsConnected(<i>NISESession sessionHandle,</i>
----------------------------------------	------------------------------------------

	<i>NISEConstString routeSpec,</i>
	<i>NISEBoolean * isConnected);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
routeSpec	NISEConstString	The string specifying the route and route groups to be checked. Refer to Route Specification Strings for more information.

Output

Name	Type	Description
isConnected	NISEBoolean	Returns TRUE if the specified route and route groups are connected or FALSE if they are not connected.

niSE_IsDebounced

Checks to see if the switching system is debounced or not. This function does not wait for debouncing to occur. It returns true if the system is fully debounced.

Function Prototype

NISEStatus __stdcall niSE_IsDebounced(<i>NISESession sessionHandle,</i>
	<i>NISEBoolean * isDebounced);</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device

		session.
--	--	----------

Output

Name	Type	Description
isDebounced	NISEBoolean	Returns TRUE if the system is fully debounced or FALSE if it is still settling.

niSE_IsDebounced Details

This function is similar to the IviSwth specific function.

niSE_OpenSession

Opens a session to a specified NI Switch Executive virtual device. Opens communications with all of the IVI switches associated with the specified NI Switch Executive virtual device. Returns a session handle that you use to identify the virtual device in all subsequent NI Switch Executive function calls.

Function Prototype

NISEStatus __stdcall niSE_OpenSession(<i>NISEConstString virtualDeviceName,</i>
	<i>NISEConstString options,</i>
	<i>NISESession * sessionHandle</i>);

Parameters

Input

Name	Type	Description
virtualDeviceName	NISEConstString	The session referencing this NI Switch Executive virtual device session.
options	NISEConstString	The option string can be used to pass information to each of the IVI devices on startup. It can be used to set things such as simulation, range checking, and so on. Consult your

		driver documentation for more information about valid entries for the option string.
--	--	--------------------------------------------------------------------------------------

Output

Name	Type	Description
<code>sessionHandle</code>	<code>NISESession</code>	The session referencing this NI Switch Executive virtual device session.

niSE_OpenSession Details

NI Switch Executive uses a reference counting scheme to manage open session handles to an NI Switch Executive virtual device. Each call to `niSE_OpenSession` must be matched with a subsequent call to `niSE_CloseSession`.

Successive calls to `niSE_OpenSession` with the same virtual device name always returns the same session handle.

NI Switch Executive disconnects its communication with the IVI switches **after** all session handles are closed to a given virtual device.

The session handles can be used safely in multiple threads of an application.

Sessions can only be opened to a given NI Switch Executive virtual device from a single process at a time.

niSE_WaitForDebounce

Waits for all of the switches in the NI Switch Executive virtual device to debounce.

Function Prototype

<code>NISEStatus __stdcall niSE_WaitForDebounce(</code>	<i>NISESession sessionHandle,</i>
	<i>NISEInt32 maxTime);</i>

Parameters

Input

Name	Type	Description
<code>sessionHandle</code>	<code>NISESession</code>	The session referencing this NI Switch Executive virtual device session.
<code>maxTime</code>	<code>NISEInt32</code>	The amount of time to wait (in milliseconds) for the debounce to complete. A value of 0 checks for debouncing once and returns an error if the system is not debounced at that time. A value of -1 means to block for an infinite period of time until the system is debounced.

niSE_WaitForDebounce Details

This function does not return until either the switching system is completely debounced and settled or the maximum time has elapsed and the system is not yet debounced.

In the event that the maximum time elapses, the function returns an error indicating that a timeout has occurred.

To ensure that all of the switches have settled, NI recommends calling `niSE_WaitForDebounce` after a series of connection or disconnection operations and before taking any measurements of the signals connected to the switching system.

NI Switch Executive Visual Basic Function Reference

Use the NI Switch Executive functions to create an application using Microsoft Visual Basic.



Note All of the following functions return `NISE_Status`. A value of 0 indicates that the function was successful. Other values represent error conditions and are available from the [Error Codes](#) table.

- [niSE_OpenSession](#)
- [niSE_CloseSession](#)
- [niSE_Connect](#)

- [niSE_ConnectAndDisconnect](#)
- [niSE_Disconnect](#)
- [niSE_DisconnectAll](#)
- [niSE_ExpandRouteSpec](#)
- [niSE_FindRoute](#)
- [niSE_GetAllConnections](#)
- [niSE_IsConnected](#)
- [niSE_IsDebounced](#)
- [niSE_WaitForDebounce](#)
- [niSE_GetIviDeviceSession](#)
- [niSE_GetError](#)
- [niSE_ClearError](#)

niSE_ClearError

Clears the error on a session.

Function Prototype

Function niSE_ClearError(<i>sessionHandle As NISESession</i>) As NISEStatus
---------------------------	------------------------------------------------------------

Parameter

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

niSE_CloseSession

Closes a session to an NI Switch Executive virtual device. Reduces the reference count of open sessions by one. If the reference count goes to 0, the function deallocates any memory resources the driver uses and closes any open IVI switch sessions.

Function Prototype

Function niSE_CloseSession(<i>sessionHandle as NISESession</i>) As NISEStatus
-----------------------------	------------------------------------------------------------

Parameter

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

niSE_CloseSession Details

After calling the niSE_CloseSession function, you should not use the NI Switch Executive virtual device again until you call [niSE_OpenSession](#).

niSE_Connect

Connects the routes specified by the connection specification. When connecting, it may allow for multiconnection based on the multiconnection mode.

Function Prototype

Function niSE_Connect(<i>sessionHandle As NISESession,</i>
	<i>connectSpec As String,</i>
	<i>multiconnectMode As NISEMulticonnectMode,</i>
	<i>waitForDebounce As NISEBoolean</i>) As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive

		virtual device session.
connectSpec	String	The string describing the connections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.
multiconnectMode	NISEMulticonnectMode	<p>This value sets the connection mode for the function. The mode may be one of the following:</p> <ul style="list-style-type: none"> • NISE_VAL_USE_DEFAULT_MODE (-1)—uses the mode selected as the default for the route in the NI Switch Executive virtual device configuration. If a mode has not been selected for the route in the NI Switch Executive virtual device, this parameter defaults to NISE_VAL_MULTICONNECT_ROUTES. • NISE_VAL_NO_MULTICONNECT (0)—routes specified in the connection specification must be disconnected before they can be reconnected. Calling <code>Connect</code> on a route that was connected using No Multiconnect mode results in an error condition. • NISE_VAL_MULTICONNECT_ROUTES (1)—routes specified in the connection specification can be connected multiple times. The first call to <code>Connect</code> performs the physical hardware connection. Successive calls to <code>Connect</code> increase a connection reference count. Similarly, calls to <code>Disconnect</code> decrease the reference count. When it reaches 0, the hardware is physically disconnected. Multiconnecting routes applies to entire routes and not to route segments.
waitForDebounce	NISEBoolean	Waits (if true) for switches to debounce between its connect and disconnect operations. If false, it immediately begins the second operation after completing the first. The order of connect and disconnect operation is set by the Operation Order input.

niSE_Connect Details

In the event of an error, the call to `niSE_Connect` attempts to undo any connections made so that the system is left in the same state that it was in before the call was made. Some errors can be caught before manipulating hardware, although it is feasible that a hardware call could fail causing some connections to be momentarily closed and then reopened.

If the wait for debounce parameter is set, the function does not return until the switch system has debounced.

niSE_ConnectAndDisconnect

Connects routes and disconnects routes in a similar fashion to `niSE_Connect` and `niSE_Disconnect` except that the operations happen in the context of a single function call. This function is useful for switching from one state to another state.

This function will not wait for debounce. Call the `niSE_WaitForDebounce` function to wait for debounce.

Function Prototype

Function <code>niSE_ConnectAndDisconnect(</code>	<i>sessionHandle As NISESession,</i>
	<i>connectSpec As String,</i>
	<i>disconnectSpec As String,</i>
	<i>multiconnectMode As NISEMulticonnectMode,</i>
	<i>operationOrder As NISEOperationOrder,</i>
	<i>waitForDebounce As NISEBoolean)</i> As NISEStatus

Parameters

Input

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

sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
connectSpec	String	The string describing the connections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.
disconnectSpec	String	The string describing the disconnections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.
multiconnectMode	NISEMulticonnectMode	<p>This value sets the connection mode for the function. The mode may be one of the following:</p> <ul style="list-style-type: none"> • NISE_VAL_USE_DEFAULT_MODE (-1)—uses the mode selected as the default for the route in the NI Switch Executive virtual device configuration. If a mode has not been selected for the route in the NI Switch Executive virtual device, this parameter defaults to NISE_VAL_MULTICONNECT_ROUTES. • NISE_VAL_NO_MULTICONNECT (0)—routes specified in the connection specification must be disconnected before they can be reconnected. Calling <code>Connect</code> on a route that was connected using No Multiconnect mode results in an error condition. • NISE_VAL_MULTICONNECT_ROUTES (1)—routes specified in the connection specification can be connected multiple times. The first call to <code>Connect</code> performs the physical hardware connection. Successive calls to <code>Connect</code> increase a connection reference count. Similarly, calls to <code>Disconnect</code> decrease the reference count. When it reaches 0, the hardware is

		<p>physically disconnected. This behavior is slightly different with SPDT relays. For more information, refer to Exclusions and SPDT Relays. Multiconnecting routes applies to entire routes and not to route segments.</p>
<p>operationOrder</p>	<p>NISEOperationOrder</p>	<p>Sets the order of the operation for the function. Defined values are Break Before Make and Break After Make.</p> <ul style="list-style-type: none"> • NISE_VAL_BREAK_BEFORE_MAKE (1)—The function disconnects the routes specified in the disconnect specification before connecting the routes specified in the connect specification. This is the typical mode of operation. <p>For NISE_VAL_BREAK_BEFORE_MAKE operation order, an asterisk (*) will disconnect all connections currently connected and connect what is in the connect specification. However, this action maintains as closed those connections currently connected that also exist in the disconnect specification. This behavior minimizes relay actuation and thus extends the life of the relays.</p> <ul style="list-style-type: none"> • NISE_VAL_BREAK_AFTER_MAKE (2)—The function connects the routes specified in the connection specification before connecting the routes specified in the disconnection specification. This mode of operation is normally used when you are switching current and want to ensure that a load is always connected to your source. The order of operation is to connect first or disconnect first. <p>For NISE_VAL_BREAK_AFTER_MAKE operation order, the VI connects what is in the connect specification and the asterisk (*) will disconnect all other connections. As in case 1, this action maintains as closed those connections currently</p>

		connected that also exist in the connect specification. This behavior minimizes relay actuation and thus extends the life of the relays.
waitForDebounce	NISEBoolean	Waits (if true) for switches to debounce between its connect and disconnect operations. If false, it immediately begins the second operation after completing the first. The order of connect and disconnect operation is set by the Operation Order input.

niSE_ConnectAndDisconnect Details

`niSE_ConnectAndDisconnect` manipulates the hardware connections and disconnections only when the routes are different between the connection and disconnection specifications. If any routes are common between the connection and disconnection specifications, NI Switch Executive determines whether the relays need to be switched. This functionality has the distinct advantage of increased throughput for shared connections, because hardware does not have to be involved and potentially increases relay lifetime by decreasing the number of times that the relay has to be switched.

In the event of an error, the call to `niSE_ConnectAndDisconnect` attempts to undo any connections made, but does not attempt to reconnect disconnections. Some errors can be caught before manipulating hardware, although it is feasible that a hardware call could fail causing some connections to be momentarily closed and then reopened.

niSE_Disconnect

Disconnects the routes specified in the Disconnection Specification.

This function will not wait for debounce. Call the `niSE_WaitForDebounce` function to wait for debounce.

Function Prototype

Function niSE_Disconnect(<i>sessionHandle As NISESession,</i>
	<i>disconnectSpec As String) As NISEStatus</i>

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
disconnectSpec	String	The string describing the disconnections to be made. The route specification strings are best summarized as a series of routes delimited by ampersands. The specified routes can be route names, route group names, or fully specified route paths delimited by square brackets. Refer to Route Specification Strings for more information.

niSE_Disconnect Details

If any of the specified routes were originally connected in a multiconnected mode, the call to `niSE_Disconnect` reduces the reference count on the route by 1. If the reference count reaches 0, it is disconnected.

If a specified route does not exist, it is an error condition.

In the event of an error, the call to `niSE_Disconnect` continues to try to disconnect everything specified by the route specification string but reports the error on completion.

niSE_DisconnectAll

Disconnects all connections on every IVI switch device managed by the NI Switch Executive session reference passed to this function.

This function will not wait for debounce. Call the `niSE_WaitForDebounce` function to

wait for debounce.

Function Prototype

Function niSE_DisconnectAll(<i>sessionHandle As NISESession</i>) As NISEStatus
------------------------------	------------------------------------------------------------

Parameter

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

niSE_DisconnectAll Details

`niSE_DisconnectAll` ignores all multiconnect modes. Calling `niSE_DisconnectAll` resets all of the switch states for the system.

niSE_ExpandRouteSpec

Expands a route spec string to yield more information about the routes and route groups within the spec.

Function Prototype

Function niSE_ExpandRouteSpec(<i>sessionHandle As NISESession,</i>
	<i>routeSpec As String,</i>
	<i>expandAction As NISEExpandAction,</i>
	<i>expandedRouteSpec As String,</i>
	<i>expandedRouteSpecSize As Long</i>) As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
routeSpec	String	The routes and route groups to be expanded. Refer to Route Specification Strings for more information.
expandAction	NISEExpandAction	<p>The mode can be one of the following:</p> <ul style="list-style-type: none"> Expand to Routes (0)— expands the route spec to routes. Converts route groups to their constituent routes. Expand to Paths (1)— expands the route spec to paths. Converts routes and route groups to their constituent square bracket route spec strings.

Output

Name	Type	Description
expandedRouteSpec	NISEBuffer*	The expanded routes and route groups. Refer to Route Specification Strings for more information.
expandedRouteSpecSize	NISEInt32	<p>The expandedRouteSpecSize is an NISEInt32 that is passed by reference into the function.</p> <p>As an input, it is the size of the route string buffer being passed. If the route string is larger than the string buffer being passed, only the portion of the route string that can fit in the string buffer is copied into it.</p> <p>On return from the function, routeSpecSize holds the size required to hold the entire route string. This size may be larger than the buffer size as the function always returns the size needed to hold the entire buffer.</p> <p>You can pass NULL for this parameter if you are not interested in the return value for routeSpecSize and routeSpec.</p>

niSE_ExpandRouteSpec Details

Use this function in conjunction with `niSE_GetAllConnections` to return the details of the currently connected routes and route groups. Then select, display, or error check what routes, devices, paths, and channels are currently connected based on the route details.

niSE_FindRoute

Finds an existing or potential route between channel 1 and channel 2.

Function Prototype

Function <code>niSE_FindRoute(</code>	<i>sessionHandle As NISESession,</i>
	<i>channel1 As String,</i>
	<i>channel2 As String,</i>
	<i>routeSpec As String,</i>
	<i>routeSpecSize As Long,</i>
	<i>routeCapability As NISERouteCapability)</i> As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
channel1	String	The channel name of one of the endpoints of the route to find. The channel name must either be a channel alias name or a name in the device/ivichannel syntax. Examples: MyChannel Switch1/R0

channel2	String	<p>The channel name of one of the endpoints of the route to find. The channel name must either be a channel alias name or a name in the device/ivichannel syntax.</p> <p>Examples: MyChannel Switch1/R0</p>
-----------------	--------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Input/Output

Name	Type	Description
routeSpecSize	Long	The routeSpecSize is a Long that is passed by reference into the function. As an input, it is the size of the route string buffer being passed. If the route string is larger than the string buffer being passed, only the portion of the route string that can fit in the string buffer is copied into it. On return from the function, routeSpecSize holds the size required to hold the entire route string. This size may be larger than the buffer size as the function always returns the size needed to hold the entire buffer. You can pass NULL for this parameter if you are not interested in the return value for routeSpecSize and routeSpec.

Output

Name	Type	Description
routeSpec	String	The fully specified route path surrounded by delimiting square brackets—if the route exists or is possible. Refer to Route Specification Strings for more information.
routeCapability	NISERouteCapability	The return value which expresses the capability of finding a valid route between Channel 1 and Channel 2. Refer to the Path Capability table for value descriptions.

niSE_FindRoute Details

The returned route specification contains the route specification, and the route capability indicates whether the route existed, is possible, or is not possible for various reasons.

The route specification string returned from `niSE_FindRoute` can be passed to

other NI Switch Executive API functions, such as `niSE_Connect`, `niSE_Disconnect`, and `niSE_ConnectAndDisconnect`, that use route specification strings.

To dynamically allocate space for the `routeSpec` buffer, call `niSE_FindRoute` twice. The first call should specify a `routeSpecSize` of 0 and then use the returned `routeSpecSize` to allocate the correct buffer size.

Path Capability

Path capability may be any one of the following:

Value	Name	Description
1	<code>NISE_VAL_PATH_AVAILABLE</code>	A path between channel 1 and channel 2 is available. The route specification parameter returns a string describing the available path.
2	<code>NISE_VAL_PATH_EXISTS</code>	A path between channel 1 and channel 2 already exists. The route specification parameter returns a string describing the existing path.
3	<code>NISE_VAL_PATH_UNSUPPORTED</code>	There is no potential path between channel 1 and channel 2 given the current configuration.
4	<code>NISE_VAL_RSRC_IN_USE</code>	There is a potential path between channel 1 and channel 2, although a resource needed to complete the path is already in use.
5	<code>NISE_VAL_EXCLUSION_CONFLICT</code>	Channel 1 and channel 2 cannot be connected because their connection would result in an exclusion violation.
6	<code>NISE_VAL_CHANNEL_NOT_AVAILABLE</code>	One of the channels is not useable as an endpoint channel. Make sure that it is not marked as a configuration channel.

`niSE_GetAllConnections`

Returns all connected routes and route groups.

Function Prototype

Function niSE_GetAllConnections(<i>sessionHandle As NISESession,</i>
	<i>routeSpec As String,</i>
	<i>routeSpecSize As Long</i>) As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

Output

Name	Type	Description
routeSpec	String	The currently connected routes and route groups. Refer to Route Specification Strings for more information.
routeSpecSize	Long	The routeSpecSize is a Long that is passed by reference into the function. As an input, it is the size of the route string buffer being passed. If the route string is larger than the string buffer being passed, only the portion of the route string that can fit in the string buffer is copied into it. On return from the function, routeSpecSize holds the size required to hold the entire route string. This size may be larger than the buffer size as the function always returns the size needed to hold the entire buffer. You can pass NULL for this parameter if you are not interested in the return value for routeSpecSize and routeSpec.

niSE_GetAllConnections Details

You can pass the routeSpec to other functions that take a routeSpec as input, such as `niSE_Connect`, `niSE_Disconnect`, and `niSE_ExpandRouteSpec`.

niSE_GetError

Queries for and returns the most recent error.

Function Prototype


Function niSE_GetError(<i>sessionHandle As NISESession,</i>
	<i>errorNumber As Long,</i>
	<i>errorDescription As String,</i>
	<i>errorDescriptionSize As Long</i>) As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

Input/Output

Name	Type	Description
errorDescriptionSize	Long	<p>The errorDescriptionSize is a Long that is passed by reference into the function.</p> <p>As an input, it is the size of the error description buffer being passed. If the error description is larger than the error description buffer being passed, only the portion of the error description that can fit in the error description buffer is copied into it.</p> <p>On return from the function, errorDescriptionSize holds the size required to hold the entire error description.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>Note errorDescriptionSize may be larger than the buffer size as the function always returns the size needed to hold the entire buffer.</p> </div>

		You can pass NULL for this parameter if you are not interested in the return value for <code>errorDescriptionSize</code> and <code>errorDescription</code> .
--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------

Output

Name	Type	Description
<code>errorNumber</code>	Long	The error code.
<code>errorDescription</code>	String	Description of the error.

To dynamically allocate space for the **errorDescription** buffer, call `niSE_GetError` twice. The first call should specify a **errorDescriptionSize** of 0 and then use the returned **errorDescriptionSize** to allocate the correct buffer size.

niSE_GetIviDeviceSession

Retrieves an IVI instrument session for an IVI switching device that is being managed by NI Switch Executive. The retrieved session handle can be used to access instrument specific functionality through the instrument driver.

The retrieved handle should **not** be closed.

Function Prototype

Function <code>niSE_GetIviDeviceSession(</code>	<i>sessionHandle As NISESession,</i>
	<i>iviLogicalName As String,</i>
	<i>iviSessionHandle As ViSession)</i> As NISEStatus

Parameters

Input

Name	Type	Description
<code>sessionHandle</code>	NISESession	The session referencing this NI Switch Executive virtual device

		session.
iviLogicalName	String	The IVI logical name of the IVI device for which to retrieve an IVI session.

Output

Name	Type	Description
iviSessionHandle	ViSession	The IVI instrument handle of the specified IVI device.

niSE_GetIviDeviceSession Details



Note Use caution when using the session handle. Calling functions on an instrument driver can invalidate the NI Switch Executive configuration and cache. You should not use the retrieved session handle to make or break connections or modify the configuration channels, as doing so can cause undefined, and potentially unwanted, behavior.

niSE_IsConnected

Checks to see whether the specified route or route group is connected.

Function Prototype

Function niSE_IsConnected(<i>sessionHandle As NISESession,</i>
	<i>routeSpec As String,</i>
	<i>isConnected As NISEBoolean)</i> As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
routeSpec	String	The routes and route groups to be checked. Refer to Route Specification Strings for more information.

Output

Name	Type	Description
isConnected	NISEBoolean	Returns TRUE if the specified route or route group is connected or FALSE if it is not connected.

niSE_IsDebounced

Checks to see if the switching system is debounced or not. This function does not wait for debouncing to occur. It returns true if the system is fully debounced.

Function Prototype

Function niSE_IsDebounced(<i>sessionHandle As NISESession,</i>
	<i>isDebounced As NISEBoolean</i>) As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

Output

Name	Type	Description
isDebounced	NISEBoolean	Returns TRUE if the system is fully debounced or FALSE if it is still settling.

niSE_IsDebounced Details

This function is similar to the IviSwtch specific function.

niSE_OpenSession

Opens a session to a specified NI Switch Executive virtual device. Opens communications with all of the IVI switches associated with the specified NI Switch Executive virtual device. Sets the configuration and source channels on each IVI device as specified by the NI Switch Executive configuration. Returns a session handle that you use to identify the virtual device in all subsequent NI Switch Executive function calls.

Function Prototype

Function niSE_OpenSession(<i>virtualDeviceName As String,</i>
	<i>optionString As String,</i>
	<i>sessionHandle As NISESession</i>) As NISEStatus

Parameters

Input

Name	Type	Description
virtualDeviceName	String	The session referencing this NI Switch Executive virtual device session.
optionString	String	The option string can be used to pass information to each of the IVI devices on startup. It can be used to set things such as simulation, range checking, and so on. Consult your driver documentation for more information about valid entries for the option string.

Output

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.

niSE_OpenSession Details

NI Switch Executive uses a reference counting scheme to manage open session

handles to an NI Switch Executive virtual device. Each call to `niSE_OpenSession` must be matched with a subsequent call to `niSE_CloseSession`.

Successive calls to `niSE_OpenSession` with the same virtual device name always returns the same session handle.

NI Switch Executive disconnects its communication with the IVI switches **after** all session handles are closed to a given virtual device.

The session handles can be used safely in multiple threads of an application.

Sessions can only be opened to a given NI Switch Executive virtual device from a single process at a time.

niSE_WaitForDebounce

Waits for all of the switches in the NI Switch Executive virtual device to debounce.

Function Prototype

Function <code>niSE_WaitForDebounce(</code>	<i>sessionHandle As NISESession,</i>
	<i>maxTime As Long</i>) As NISEStatus

Parameters

Input

Name	Type	Description
sessionHandle	NISESession	The session referencing this NI Switch Executive virtual device session.
maxTime	Long	The amount of time to wait (in milliseconds) for the debounce to complete. A value of 0 checks for debouncing once and returns an error if the system is not debounced at that time. A value of -1 means to block for an infinite period of time until the system is debounced.

niSE_WaitForDebounce Details

This function does not return until either the switching system is completely debounced and settled or the maximum time has elapsed and the system is not yet debounced.

In the event that the maximum time elapses, the function returns an error indicating that a timeout has occurred.

To ensure that all of the switches have settled, NI recommends calling `niSE_WaitForDebounce` after a series of connection or disconnection operations and before taking any measurements of the signals connected to the switching system.

Error Codes



Note All of the following error codes begin with `NISE_ERROR_`.

Warning Code	Hex Value	Decimal Value	Description
INTERNAL	FFFF8EB8	-29000	An internal error has occurred. Please contact National Instruments technical support.
IVI_DRIVER_NO_SIMULATION	FFFF8EB7	-29001	The IVI-specific driver does not support simulation mode.
INVALID_VIRTUAL_DEVICE_NAME	FFFF8EB6	-29002	The specified NI Switch Executive virtual device is invalid or does not exist.
INVALID_SESSION	FFFF8EB5	-29003	This session is not a valid NI Switch Executive virtual device session.
INSUFFICIENT_SYSTEM_RESOURCES	FFFF8EB4	-29004	NI Switch Executive requires system resources that are currently unavailable. Close other applications and try again.
AMBIGUOUS_NAME	FFFF8EB3	-29005	The specified name is ambiguous. Specify a unique name.

INVALID_IVI_LOGICAL_NAME	FFFF8EB2	-29006	The IVI logical name is invalid or the device does not exist.
INVALID_ROUTE_SPECIFICATION	FFFF8EB1	-29007	The route specification string contains invalid characters or could not be understood.
EVAL_TIMED_OUT	FFFF8EB0	-29008	NI Switch Executive is running with an evaluation license and the time limit for this session has expired. Restart your application to continue evaluating.
INVALID_NAME	FFFF8EAF	-29009	The name contains invalid characters.
RUNTIME_IMPORTING_EVAL	FFFF8EAE	-29010	This configuration was created with an evaluation license. Deployment licenses can only import configurations created with a development license. Import the configuration into a development license system and re-export the file to resolve the problem.
EVAL_EXPIRED_IMPORTING	FFFF8EAD	-29011	NI Switch Executive cannot import configurations when the evaluation period has expired.
RUNTIME_EXPORTING	FFFF8EAC	-29012	NI Switch Executive deployment licenses cannot export configurations. The operation requested requires a development license.
IMPORTING_FILE_ACCESS	FFFF8EAB	-29013	There was an error when accessing (open or read) the NI Switch Executive configuration file.
IMPORTING_FILE_FORMAT	FFFF8EAA	-29014	The file is not a valid NI Switch Executive configuration file.
INVALID_END_POINTS	FFFF8EA9	-29015	The endpoints of the path do not match the existing endpoints.

INVALID_PATH	FFFF8EA8	-29016	Cannot connect this path on the device.
INVALID_CHANNEL_SPECIFICATION	FFFF8EA7	-29017	The channel specification string contains invalid characters or could not be understood.
DLL_NOT_FOUND	FFFF8EA6	-29018	A needed DLL was not found. Check to ensure that NI Switch Executive is properly installed and that all needed DLLs are in the search path.
FUNCTION_NOT_FOUND	FFFF8EA5	-29019	A needed function in a DLL could not be found. Although the DLL exists, it may be an incorrect version and may not contain the needed function.
MAX_TIME_EXCEEDED	FFFF8EA4	-29020	One or more switching devices have not debounced within the specified maximum time.
ROUTE_ALREADY_EXISTS	FFFF8EA3	-29021	The route you are trying to connect or a route with the same endpoints is already connected.
ROUTE_EXISTS_AS_UNSHAREABLE	FFFF8EA2	-29022	The route you are trying to connect already exists as a non-multiconnect route. It must be disconnected before you can make a multiconnect route.
ROUTE_EXISTS_BY_DIFFERENT_PATH	FFFF8EA1	-29023	The route you are trying to connect or disconnect already exists but is connected through a different path than the one specified.
ROUTE_DOES_NOT_EXIST	FFFF8EA0	-29024	The specified route does not exist. You cannot disconnect a route that does not exist.
PARTIAL_DISCONNECT	FFFF8E9F	-29025	Device specific errors occurred during the disconnect operation.
RESOURCE_IN_USE	FFFF8E9E	-29026	A connection could not be made because one of the switch

			resources needed to make the connection is being used as part of another currently connected route.
FILE_WRITE	FFFF8E9D	-29027	An error occurred while attempting to write to file.
FILE_READ	FFFF8E9C	-29028	An error occurred while attempting to read from file.
INVALID_MULTICONNECT_MODE	FFFF8E9B	-29029	Invalid multiconnect mode.
INVALID_OPERATION_ORDER	FFFF8E9A	-29030	Invalid operation order.
CONFIG_CHANNEL_CONFLICT	FFFF8E99	-29031	A reserved for routing channel required for connecting this route is already in use by another route.
SOURCE_CHANNEL_CONFLICT	FFFF8E98	-29032	Connecting this route would cause excluded channels to be shorted together.
ROUTE_EXISTS_WITH_DIFFERENT_MODE	FFFF8E97	-29033	The route you are trying to connect already exists with a different multiconnect mode. It must be disconnected before you can make this connection.
DISABLED_CHANNEL	FFFF8E96	-29034	The channel you are trying to use has been disabled for this virtual device.
CANNOT_CONNECT_TO_ITSELF	FFFF8E95	-29035	You cannot connect a channel to itself.
ROUTE_NOT_FOUND	FFFF8E94	-29036	Route cannot be found between the specified endpoints. Either your endpoint channels are the same, or they reside on the same hardware.

Examples

NI Switch Executive programming examples are instructional tools that demonstrate how to integrate NI Switch Executive virtual devices into your systems. NI

Switch Executive programming examples are available for the following ADEs:

- LabVIEW
- LabWindows™/CVI™
- C/C++
- Visual Basic
- NI TestStand

For example locations and supported ADE versions, refer to the [NI Switch Executive Readme](#) file.

Frequently Asked Questions

- [Why do all of my routes use the same reserved for routing channel?](#)
- [How do I deploy my NI Switch Executive configuration to target systems?](#)
- [How do I create a simulated NI-SWITCH device and use it in my application?](#)
- [How do I create a switch logical name from MAX?](#)
- [How do I use my third-party switch?](#)

Why do all of my routes use the same reserved for routing channel?

Routes are created independently and therefore have no knowledge of what routing channels are in use by other routes. By placing routes into a [route group](#), NI Switch Executive configures the routes so that the routes do not use the same resources and can coexist. To force routes to use different routing channel(s), make them members of the same route group.

How do I deploy my NI Switch Executive configuration to target systems?

Refer to [Deploying a Virtual Device](#) for more information.

How do I create a simulated NI-SWITCH device and use it in my application?

Simulated switches are useful for creating your NI Switch Executive virtual device without physical hardware. To create a simulated switch, complete the following steps:

1. Launch Measurement and Automation Explorer (MAX).
2. Right-click **Devices and Interfaces** and select **Create New**.
3. Select **Simulated NI-DAQmx Device or Modular Instrument** and click **Finish**.
4. Expand **Switches**, select the switch you want to simulate, and click **OK**. The simulated device appears under Devices and Interfaces in the MAX tree.
5. Right-click the simulated device in the MAX Tree and select **Rename** to rename the simulated device.
6. Configure the simulated device.
 - To configure a non-NI SwitchBlock simulated device, complete the following steps:
 1. Right-click the simulated device in the MAX Tree and select **Configure**. The Device Properties dialog box appears.
 2. Select a terminal block and topology from the drop-down listboxes in the **Terminal Block/Topology** tab.
 3. Select channels and reserve channels for routing in the **Channels** tab.
 4. Click **OK** to apply your changes and reset the device.
 - To configure an NI SwitchBlock simulated device, complete the following steps:
 1. Right-click the simulated carrier in the MAX tree and select **Insert or Remove Simulated Cards** to add or remove cards in a carrier.
 2. Click **OK** to save changes.
 3. Right-click the simulated carrier in the MAX tree and select **Configure** to combine and uncombine cards in a carrier.
 4. Click **OK** to save changes.

How do I create a switch logical name from MAX?

Refer to **Create a Logical Name** of [Using Third-Party Switches](#) for information about how to create a switch logical name from MAX.

How do I use my third-party switch?

Refer to [Using Third-Party Switches](#) for information about using third-party switches with NI Switch Executive. For more information about adding IVI switches to an NI Switch Executive virtual device, refer to [Adding IVI Switches](#).
