IVI-4.6: IviSwtch Class Specification

September 24, 2015 Edition
Revision 4.0

Important Information

The IviSwtch Class Specification (IVI-4.6) is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at www.ivifoundation.org.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through the web site at www.ivifoundation.org.

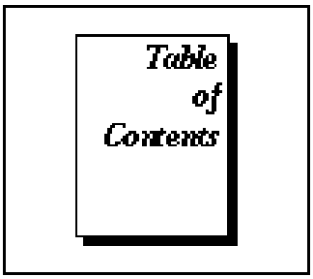
Warranty

The IVI Foundation and its member companies make no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The IVI Foundation and its member companies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Trademarks

Product and company names listed are trademarks or trade names of their respective companies.

No investigation has been made of common-law trademark rights in any work.



IviSwtch Class Specification

IviSwtch Revision History

This section is an overview of the revision history of the IviSwtch specification.

Table 1. IviSwtch Class Specification Revisions

Revision Number	Date of Revision	Revision Notes
Revision 1.0b1	June 26, 1998	First Approved Version.
Revision 1.1	August 21, 1998	Technical Publications review and edit. Changes to template information.
Revision 2.0	November 22, 1999	Refined the organization of the specification based on feedback at the July 1999 IVI Foundation meeting.
Revision 2.0a	May 25, 2001	First draft to include COM requirements. Added timeout errors for Wait...() functions.
Revision 2.1vc1	July 30, 2001	Voting candidate 1. This revision adds functions and attributes for cross class capabilities, the standard IVI-C header file and revised IDL files. C hierarchies were updated. There are also several spelling, wording, and syntax corrections.
Revision 2.1vc2	October 30, 2001	Voting Candidate 2. Improved the description of some attributes. Removed inherent capabilities from hierarchy tables. IDL checked for consistency and updated. Added text referring to COM compliance notes for attribute values. Added table with error message strings. Added Max Time Exceeded error code. Added text describing the repeated capabilities. Other minor style changes.
Revision 2.1vc3	December 20, 2001	Voting Candidate 3. Get Channel Name C function separated from Name COM attribute. Other changes according to the outcome of the December meeting (see minutes) Updated for consistency with revised IVI-3.1. Minor style updates.
Revision 2.1vc4	January 3, 2002	Voting Candidate 4. Changed “Applies To” for Channel Count attribute.
Revision 3.0 vc5	February 4, 2002	Voting Candidate 5. Changed version to 3.0. Updates from review feedback.
Revision 3.0 vc6	February 5, 2002	Voting Candidate 6. Minor correction to text in Section 4.1.
Revision 3.0 vc7	March 4, 2002	Voting Candidate 7. Included IDL for final version of COM type libraries. Changed MaxTime to

Table 1. IviSwtch Class Specification Revisions

Revision Number	Date of Revision	Revision Notes
		MaxTimeMilliseconds.
Revision 3.0	April 12, 2002	Released version 3.0, including the COM interface specification. No content change from Voting Candidate 7.
Revision 3.0	April 29, 2008	Editorial change to update the IVI Foundation contact information in the Important Information section to remove obsolete address information and refer only to the IVI Foundation web site.
Revision 3.0	April 2009	Editorial change to update repeated capabilities section to include both qualified and unqualified repeated capability names.
Revision 3.0	April 28, 2009	Minor change to update IviSwtch_SetPath function description with additional possible values for channel names in the path string. (Section 4.3.8.) Editorial change to add more specific information in the driver development guidelines for general purpose switches. (Appendix A)
Revision 4.0	June 9, 2010	Incorporated IVI.NET
Revision 4.0	August 25, 2011	Editorial IVI.NET change. Change references to process-wide locking to AppDomain-wide locking. Add an overload to the Create factory method that takes locking related parameters.
Revision 4.0	March10, 2012	Editorial Change: Delete InvalidSwitchPathException (not needed and never implemented) from section 4.3.8 and correct two API spelling errors.
Revision 4.0	August 6, 2012	Editorial Changes: Correct the description of the NoSuchPathException in section 10.1.13.
Revision 4.0	June 21, 2013	Editorial Changes: Remove the index parameter from the parameter table in section 4.2.8. In section 4.2.9, make it explicit that the 1-based index only applies to COM.
Revision 4.0	September 24, 2015	Editorial Change – Clarified the use of one-based index for COM, and zero-based index for .NET for repeated capabilities in section 4.2.9.

API Versions

Architecture	Drivers that comply with version 4.0 comply with all of the versions below

C	2.0, 3.0, 4.0
COM	3.0, 4.0
.NET	4.0

Drivers that comply with this version of the specification also comply with earlier, compatible, versions of the specification as shown in the table above. The driver may benefit by advertising that it supports all the API versions listed in the table above.

1 Overview of the IviSwtch Specification

1.1 Introduction

This specification defines the IVI class for switches. The IviSwtch class is designed to support the typical switches as well as common extended functionality found in specialized switch modules. This section summarizes the *IviSwtch Class Specification* itself and contains general information that the reader may need in order to understand, interpret, and implement aspects of this specification. These aspects include the following:

- IviSwtch Class Overview
- The definitions of terms and acronyms
- References

1.2 IviSwtch Class Overview

This specification describes the IVI class for switches. The IviSwtch class is designed to support the typical switches as well as common extended functionality found in specialized switch modules.

An IviSwtch is a vendor-defined *switch module* with a series of I/O capable *channels*. These channels can then be connected through the internals of the switch module, where not all connections are necessarily valid. An example is shown below in Figure 1. The IviSwtch class conceptualizes the switch as an instrument that can establish paths between its I/O channels.

The IviSwtch class is divided into a base capability group and multiple extension groups. The base capability group is used to create and destroy paths on a typical switch module, and to determine if the creation of a path is possible between two switch I/O channels. The IviSwtch base capability group is described in Section 4, *IviSwtchBase Capability Group*.

In addition to the base capability group, the IviSwtch class defines extended capability groups for switches that can wait for the trigger to establish or break paths on the switch module, and assert a trigger after an operation is complete The switches that can perform such tasks are the part of the IviSwtchScanner extension group.

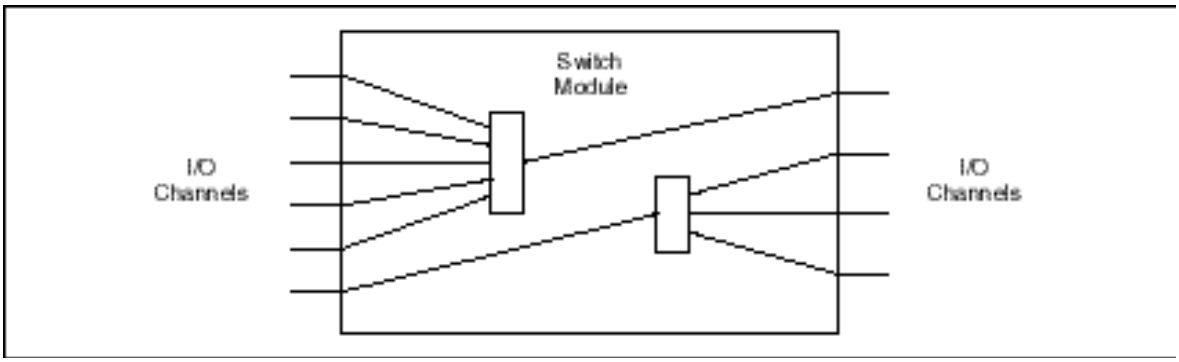


Figure 1-1. Switch Module

1.3 References

Several other documents and specifications are related to this specification. These other related documents are the following:

- IVI-3.1: Driver Architecture Specification
- IVI-3.2: Inherent Capabilities Specification
- IVI-3.3: Standard Cross Class Capabilities
- IVI-3.4: API Style Guide
- IVI-3.18: IVI.NET Utility Classes and Interfaces Specification
- IVI-5.0: Glossary

1.4 Definitions of Terms and Acronyms

This section defines terms and acronyms that are specific to the IviSwtch class. Terms of more general interest are defined in *IVI-5.0: Glossary*.

Channel	An input/output (I/O) connection on the switch module that a user can access. What constitutes a channel is up to the vendor, but the channel must be a point that you can connect to one or more other channels of the switch module through a path. In addition, a channel is the connection point to the switch module. Notice that a channel does not indicate the number of wires. A channel may consist of 1, 2, 3 or 4 wires, for example.
Channel Pair	Two channel names separated by the "->"symbol.
Common	The name of the output channel in a multiplexer switch module.
Configuration Channel	A channel that is either not directly accessible to the user through the IviSwtch class driver, or a channel that the user marks as a configuration channel reserved for path creation. The driver uses a configuration channel to create paths between the channels, connect or disconnect to an analog bus, etc. This gives the driver more flexibility in creating paths at the expense of losing channels. Mark a column in a matrix as a configuration channel when you want to allow the matrix to connect a row to a row.
Matrix Switch Module	A switch module that is configured to have multiple inputs and outputs that form a standard matrix organization such that any row can be connected to any column. Notice that some, but not all matrices support row-to-row and column-to-column connections. See Configuration Channel.
Multiplexer Switch Module	A switch module that is configured to have multiple input channels but only a single output channel. Other names for the multiplexer switch module are "tree" and "1×n matrix."

Path	The connection (electrical, optical, etc.) between the two channels. You create a path with operations defined in the IviSwtch class. The end-point channels define such a connection. Notice that it is up to the switch module to know what paths are valid, invalid or in use.
Scanner Switch Module	An IviSwtch switch module with the capability to scan channels.
Source Channel	A channel directly accessible by the user through the IviSwtch class driver. Typically, the driver marks a channel as a source channel to allow for external connection.
Switch Module	The vendor defined device that the instrument driver session can communicate with and control. The channels of such a device define a switch module. Notice that on a physical switch card there may be multiple switch modules. In addition, a switch module may be on multiple switch cards. The concept is to have a single black box with external connections and have the software find the necessary paths. Notice that this does not remove the need of the application programmer to understand the underlying switch structure and recognize issues such as sending the correct signals through the correct switches (for example, RF signals through RF paths only).
UUT	Unit Under Test.

2 IviSwtch Class Capabilities

2.1 Introduction

The IviSwtch specification divides switch capabilities into a base capability group and multiple extension capability groups. Each capability group is discussed in a separate section. This section defines names for each capability group and gives an overview of the information presented for each capability group.

2.2 IviSwtch Group Names

The capability group names for the IviSwtch class are defined in the following table. The group name is used to represent a particular capability group and is returned as one of the possible group names from the Class Group Capabilities attribute.

Table 2-1. IviSwtch Group Names

Group Name	Description
`IviSwtchBase`	Base capabilities of the IviSwtch specification. This group supports the ability to connect and disconnect paths on the instrument, determine the connectivity of two switches, and query the state of the switch module.
`IviSwtchScanner`	This group supports the IviSwtchBase capabilities and has the ability to scan channels.
`IviSwtchSoftwareTrigger`	This group supports the IviSwtchBase capabilities and has the ability to receive software triggers.

2.3 IviSwtch Repeated Capability Names

The IviSwtch specification defines one repeated capability:

- Channel

Refer to the sections of *IVI-3.1, Driver Architecture Specification* that deal with repeated capabilities. The relevant sections are Section 2.7, *Repeated Capabilities*, Section 4.1.9, *Repeated Capabilities*, Section 4.2.5, *Repeated Capabilities*, Section 4.3.9, *Repeated Capabilities*, and Section 5.9, *Repeated Capability Identifiers and Selectors*.

2.3.1 Channel

In the configuration store, the name for the channel repeated capability shall be exactly one of "Channel" or "IviSwtchChannel". Drivers that implement multiple repeated capabilities with the name "channel" shall use the latter form to disambiguate the names.

2.4 Boolean Attribute and Parameter Values

This specification uses True and False as the values for Boolean attributes and parameters. The following table defines the identifiers that are used for True and False in the IVI.NET, IVI-COM, and IVI-C architectures.

Boolean Value	IVI.NET Identifier	IVI-COM Identifier	IVI-C Identifier
True	`true`	`VARIANT_TRUE`	`VI_TRUE`
False	`false`	`VARIANT_FALSE`	`VI_FALSE`

2.5 .NET Namespace

The .NET namespace for the IviSwtch class is `Ivi.Swtch`.

2.6 .NET IviSwtch Session Factory

The IviSwtch .NET assembly contains a factory method called Create for creating instances of IviSwtch class-compliant IVI.NET drivers from driver sessions and logical names. Create is a static method accessible from the static IviSwtch class.

Refer to *IVI-3.5: Configuration Server Specification* for a description of how logical names and session names are defined in the configuration store.

Refer to Section 8, *IVI.NET Specific Driver Constructor*, of *IVI-3.2: Inherent Capabilities Specification*, for more details on how the `idQuery`, `reset`, and `options` parameters affect the instantiation of the driver.

Refer to Section 4.3.11, *Multithread Safety*, of *IVI-3.1: Driver Architecture Specification* for a complete description of IVI.NET driver locking. Refer to Section 8, Table 8.2 *Required Lock Type Behavior for Drivers With the Same Access Key*, of *IVI-3.2, Inherent Capability Specification*, for an explanation of how the values for `lockType` and `accessKey` are used to determine the kind of multithreaded lock to use for the driver instance.

.NET Method Prototype

```
IIviSwtch Ivi.Swtch.Create(String name);

IIviSwtch Ivi.Swtch.Create(String name,
                           Boolean idQuery,
                           Boolean reset);

IIviSwtch Ivi.Swtch.Create(String name,
                           Boolean idQuery,
                           Boolean reset,
                           String options);

IIviSwtch Ivi.Swtch.Create(String resourceName,
                           Boolean idQuery,
                           Boolean reset,
                           LockType lockType,
                           String accessKey,
```

```
String options);
```

Parameters

Inputs	Description	Base Type
name	A session name or a logical name that points to a session that uses an IVI.NET IviSwtch class-compliant driver.	String
idQuery	Specifies whether to verify the ID of the instrument. The default is False.	Boolean
reset	Specifies whether to reset the instrument. The default is False.	Boolean
lockType	Specifies whether to use AppDomain-wide locking or machine-wide locking.	Ivi.Driver.LockType
accessKey	Specifies a user-selectable access key to identify the lock. Driver instances that are created with the same accessKey will be protected from simultaneous access by multiple threads within an AppDomain or across AppDomains, depending upon the value of the lockType parameter.	String
options	A string that allows the user to specify the initial values of certain inherent attributes. The default is an empty string.	String

Outputs	Description	Base Type
Return Value	Interface pointer to the IIviSwtch interface of the driver referenced by session.	IIviSwtch

Defined Values

Name	*Description*		
		Language	*Identifier*
AppDomain	The lock is AppDomain-wide.		
		.NET	Ivi.Driver.LockType.AppDomain
Machine	The lock is machine-wide.		
		.NET	Ivi.Driver.LockType.Machine

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Usage

To create a driver that implements the IviSwtch instrument class API from the logical name "My LogicalName" use the following code:

```
IIviSwtch swtch = IviSwtch.Create("MyLogicalName");
```

In this case, the ID of the instrument will not be verified, the instrument will not be reset, and options will be supplied from the configuration store and/or driver defaults.

3 General Requirements

This section describes the general requirements a specific driver shall meet in order to be compliant with this specification. In addition, it provides general requirements that specific drivers shall meet in order to comply with a capability group, attribute, or function.

3.1 Minimum Class Compliance

To be compliant with the IviSwtch Class Specification, a specific driver shall conform to all of the requirements for an IVI class-compliant specific driver specified in *IVI-3.1: Driver Architecture Specification*, implement the inherent capabilities that *IVI- 3.2: Inherent IVI Capabilities Specification* defines and implement the IviSwtchBase capability group.

3.1.1 Disable

Refer to *IVI-3.2: Inherent Capabilities Specification* for the prototype of this function.

The Disable function shall cause the Switch to disconnect all paths, if the switch module allows this operation. Notice that some switch modules may not be able to disconnect all paths (such as a scanner that must keep at least one path).

3.2 Capability Group Compliance

IVI-3.1: Driver Architecture Specification defines the general rules for a specific driver to be compliant with a capability group.

4 IviSwtchBase Capability Group

4.1 IviSwtchBase Overview

The IviSwtchBase Capability Group defines attributes and their values to determine the characteristics of I/O channels and the status of paths. The IviSwtchBase Capability Group also includes functions for creating and destroying paths on a switch module, and for determining if the creation of a path is possible between two I/O channels.

4.2 IviSwtchBase Attributes

The IviSwtchBase capability group defines the following attributes:

- AC Current Carry Max
- AC Current Switching Max
- AC Power Carry Max
- AC Power Switching Max
- AC Voltage Max
- Bandwidth
- Channel Count
- Channel Item (COM and .NET only)
- Channel Name (COM and .NET only)
- Characteristic Impedance
- DC Current Carry Max
- DC Current Switching Max
- DC Power Carry Max
- DC Power Switching Max
- DC Voltage Max
- Is Configuration Channel
- Is Debounced
- Is Source Channel
- Settling Time
- Wire Mode

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 7, *IviSwtch Attribute ID* Definitions.

4.2.1 AC Current Carry Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.ACCurrentCarryMax
```

COM Property Name

```
Channels.Item().Characteristics.ACCurrentCarryMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_CARRY_AC_CURRENT
```

Description

The maximum AC current the channel can carry, in amperes RMS.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.2 AC Current Switching Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.ACCurrentSwitchingMax
```

COM Property Name

```
Channels.Item().Characteristics.ACCurrentSwitchingMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_SWITCHING_AC_CURRENT
```

Description

The maximum AC current the channel can switch, in amperes RMS.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.3 AC Power Carry Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.ACPowerCarryMax
```

COM Property Name

```
Channels.Item().Characteristics.ACPowerCarryMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_CARRY_AC_POWER
```

Description

The maximum AC power the channel can handle, in volt-amperes.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.4 AC Power Switching Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.ACPowerSwitchingMax
```

COM Property Name

```
Channels.Item().Characteristics.ACPowerSwitchingMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_SWITCHING_AC_POWER
```

Description

The maximum AC power the channel can switch, in volt-amperes.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.5 AC Voltage Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.ACVoltageMax
```

COM Property Name

```
Channels.Item().Characteristics.ACVoltageMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_AC_VOLTAGE
```

Description

The maximum AC voltage the channel can handle, in volts RMS.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.6 Bandwidth

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.Bandwidth
```

COM Property Name

```
Channels.Item().Characteristics.Bandwidth
```

C Constant Name

```
IVISWTCH_ATTR_BANDWIDTH
```

Description

The maximum frequency signal, in Hertz, that can pass through the channel. without attenuating it by more than 3dB.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.7 Channel Count

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	RO	Channel	N/A	None

.NET Property Name

Channels.Count

This property is inherited from IIviRepeatedCapabilityCollection.

COM Property Name

Channels.Count

C Constant Name

IVISWTCH_ATTR_CHANNEL_COUNT

Description

Returns the number of available channels.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.8 Channel Item (COM and .NET only)

Data Type	Access	Applies to	Coercion	High Level Functions
IIviSwtchChannel*	RO	Channel	N/A	None

.NET Property Name

```
Channels[String name]
```

This indexer is inherited from `IIviRepeatedCapabilityCollection`. The string parameter uniquely identifies a particular channel in the channels collection.

COM Property Name

```
Channels.Item ([in] BSTR Name);
```

C Constant Name

N/A

Description

Channel Item uniquely identifies a channel in the channels collection. It returns an interface pointer which can be used to control the attributes and other functionality of that channel.

The Item property takes a channel name. If the user passes an invalid value for the source name parameter, the property returns an error.

Valid names include physical repeated capability identifiers and virtual repeated capability identifiers.

Parameters

Inputs	Description	Datatype
name (.NET) Name (COM)	Specifies the name of the channel to retrieve.	ViConstString

Return Values (C/COM)

If the IVI-COM driver cannot recognize the `Name` parameter, it returns an Unknown Name in Selector completion code as described in *IVI-3.2: Inherent Capabilities Specification*, Section 9.3.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this indexer.

4.2.9 Channel Name (COM and .NET only)

Data Type	Access	Applies to	Coercion	High Level Functions
ViString	RO	Channel	N/A	GetChannelName (C Only)

.NET Property Name

```
Channels[].Name
```

This property is inherited from `IIviRepeatedCapabilityIdentification`.

COM Property Name

```
Channels.Name([in] LONG Index);
```

C Constant Name

N/A

(Use the `GetChannelName` function.)

Description

This attribute returns the physical name identifier defined by the specific driver for the Channel.

In COM, this name corresponds to the one-based index that the user specifies. In .NET, the index is zero-based. If the driver defines a qualified channel name, this property returns the qualified name. If the value that the user passes for the `Index` parameter is less than one or greater than the value of the Channel Count, the attribute returns an empty string for the value and returns an error.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.10 Characteristic Impedance

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.Impedance
```

COM Property Name

```
Channels.Item().Characteristics.Impedance
```

C Constant Name

```
IVISWTCH_ATTR_CHARACTERISTIC_IMPEDANCE
```

Description

The characteristic impedance of the channel, in ohms.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.11 DC Current Carry Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.DCCurrentCarryMax
```

COM Property Name

```
Channels.Item().Characteristics.DCCurrentCarryMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_CARRY_DC_CURRENT
```

Description

The maximum DC current the channel can carry, in amperes.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.12 DC Current Switching Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.DCCurrentSwitchingMax
```

COM Property Name

```
Channels.Item().Characteristics.DCCurrentSwitchingMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_SWITCHING_DC_CURRENT
```

Description

The maximum DC current the channel can switch, in amperes.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.13 DC Power Carry Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.DCPowerCarryMax
```

COM Property Name

```
Channels.Item().Characteristics.DCPowerCarryMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_CARRY_DC_POWER
```

Description

The maximum DC power the channel can handle, in watts.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.14 DC Power Switching Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.DCPowerSwitchingMax
```

COM Property Name

```
Channels.Item().Characteristics.DCPowerSwitchingMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_SWITCHING_DC_POWER
```

Description

The maximum DC power the channel can switch, in watts.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.15 DC Voltage Max

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channel	N/A	None

.NET Property Name

```
Channels[].Characteristics.DCVoltageMax
```

COM Property Name

```
Channels.Item().Characteristics.DCVoltageMax
```

C Constant Name

```
IVISWTCH_ATTR_MAX_DC_VOLTAGE
```

Description

The maximum DC voltage the channel can handle, in volts.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.16 Is Configuration Channel

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

.NET Property Name

```
Channels[].IsConfigurationChannel
```

COM Property Name

```
Channels.Item().IsConfigurationChannel
```

C Constant Name

```
IVISWTCH_ATTR_IS_CONFIGURATION_CHANNEL
```

Description

Specifies whether the specific driver uses the channel for internal path creation. If set to True, the channel is no longer accessible to the user and can be used by the specific driver for path creation. If set to False, the channel is considered a standard channel and can be explicitly connected to another channel.

For example, if the user specifies a column-to-column connection in a matrix, it typically must use at least one row channel to make the connection. Specifying a channel as a configuration channel allows the instrument driver to use it to create the path.

Notice that once a channel has been configured as a configuration channel, then no operation can be performed on that channel, except for reading and writing the Is Configuration Channel attribute.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.17 Is Debounced

Data Type	Access	Applies to	Coercion	High Level Functions
ViBoolean	RO	N/A	N/A	Is Debounced

.NET Property Name

```
Path.IsDebounced
```

COM Property Name

```
Path.IsDebounced
```

C Constant Name

```
IVISWTCH_ATTR_IS_DEBOUNCED
```

Description

This attribute indicates whether the switch module has settled from the switching commands and completed the debounce. If True, the switch module has settled from the switching commands and completed the debounce.It indicates that the signal going through the switch module is valid, assuming that the switches in the path have the correct characteristics. If False, the switch module has not settled.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.18 Is Source Channel

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

.NET Property Name

```
Channels[].IsSourceChannel
```

COM Property Name

```
Channels.Item().IsSourceChannel
```

C Constant Name

```
IVISWTCH_ATTR_IS_SOURCE_CHANNEL
```

Description

Allows the user to declare a particular channel as a source channel. If set to True, the channel is a source channel. If set to False, the channel is not a source channel.

If a user ever attempts to connect two channels that are either sources or have their own connections to sources, the path creation operation returns an error. Notice that the term source can be from either the instrument or the UUT perspective. This requires the driver to ensure with each connection that another connection within the switch module does not connect to another source.

The intention of this attribute is to prevent channels from being connected that may cause damage to the channels, devices, or system. Notice that GROUND can be considered a source in some circumstances.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.19 Settling Time

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64 (C/COM)	RO	Channel	N/A	None
PrecisionTimeSpan (.NET)	RO	Channel	N/A	

.NET Property Name

```
Channels[].Characteristics.SettlingTime
```

COM Property Name

```
Channels.Item().Characteristics.SettlingTime
```

C Constant Name

```
IVISWTCH_ATTR_SETTLING_TIME
```

Description

The maximum total settling time for the channel before the signal going through it is considered stable. This includes both the activation time for the channel as well as any debounce time.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

For C and COM, time is in seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.20 Wire Mode

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	RO	Channel	None	None

.NET Property Name

```
Channels[].Characteristics.WireMode
```

COM Property Name

```
Channels.Item().Characteristics.WireMode
```

C Constant Name

```
IVISWTCH_ATTR_WIRE_MODE
```

Description

This attribute describes the number of conductors in the current channel.

Notice that values for this attribute are on per-channel basis and may not take into account the other switches that make up a path to or from this channel.

For example, this attribute returns 2 if the channel has two conductors.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3 IviSwtchBase Functions

The IviSwtchBase capability group defines the following functions:

- Can Connect
- Connect
- Disconnect
- Disconnect All
- Get Channel Name (IVI-C only)
- Get Path
- Is Debounced (IVI-C only)
- Set Path
- Wait For Debounce

This section describes the behavior and requirements of each function.

4.3.1 Can Connect

Description

The purpose of this function is to allow the user to verify whether the switch module can create a given path without the switch module actually creating the path. In addition, the operation indicates whether the switch module can create the path at the moment based on the current paths in existence.

Notice that while this operation is available for the end user, the primary purpose of this operation is to allow higher-level switch drivers to incorporate IviSwtch drivers into higher level switching systems.

If the implicit connection exists between the two specified channels, this functions returns the warning Implicit Connection Exists.

.NET Prototype

```
Ivi.Swtch.PathCapability Path.CanConnect(String channel1,
                                         String channel2);
```

COM Prototype

```
HRESULT Path.CanConnect([in] BSTR Channel1,
                        [in] BSTR Channel2,
                        [out,retval] IviSwtchPathCapabilityEnum
                          *PathCapability);
```

C Prototype

```
ViStatus IviSwtch_CanConnect (ViSession Vi, ViConstString Channel1,
                              ViConstString Channel2, ViInt32 *PathCapability);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession
Channel1	A string indicating one of the channels of the path.	ViConstString
Channel2	A string indicating one of the channels of the path.	ViConstString

Outputs	Description	Data Type
PathCapability (C/COM)	Indicates whether a path is valid and/or possible. See below for definitions.	ViInt32
Return Value (.NET)	Indicates whether a path is valid and/or possible. See below for definitions.	ViInt32

Defined Values for PathCapability Parameter

Name	*Description*		
		Language	*Identifier*
Path Available	The driver can create a path at this time.		
		.NET	PathCapability.Available

Name	*Description*		
		Language	*Identifier*
		C	IVISWTCH_VAL_PATH_AVAILABLE
		COM	IviSwtchPathAvailable
Path Exists	The explicit path between the channels already exists.		
		.NET	PathCapability.Exists
		C	IVISWTCH_VAL_PATH_EXISTS
		COM	IviSwtchPathExists
Path Unsupported	The instrument is not capable of creating a path between the two channels.		
		.NET	PathCapability.Unsupported
		C	IVISWTCH_VAL_PATH_UNSUPPORTED
		COM	IviSwtchPathUnsupported
Resource In Use	Although the path is valid, the driver cannot create the path at this moment because the switch module is currently using one or more of the required channels to create another path. You must destroy the other path before creating this one.		
		.NET	PathCapability.ResourceInUse
		C	IVISWTCH_VAL_RSRC_IN_USE
		COM	IviSwtchPathRsrcInUse
Source Conflict	The instrument cannot create a path between the two channels because both are connected to a different source channel.		
		.NET	PathCapability.SourceConflict
		C	IVISWTCH_VAL_SOURCE_CONFLICT
		COM	IviSwtchPathSourceConflict
Channel Not Available	The driver cannot create a path between the two channels because one of the channels is a configuration channel and thus unavailable for external connections.		
		.NET	PathCapability.ChannelNotAvailable
		C	IVISWTCH_VAL_CHANNEL_NOT_AVAILABLE
		COM	IviSwtchPathChannelNotAvailable

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Implicit Connection Exists	Warning: The implicit connection exists between the channels.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined warning events for this method.

Warning	Description
Implicit Connection Exists	The implicit connection exists between the channels.

Compliance Notes

1. If an IVI-C specific driver defines additional values for the `PathCapability` parameter, the actual values shall be greater than or equal to `IVISWTCH_VAL_CAN_CONNECT_SPECIFIC_EXT_BASE`.

2. If an IVI-C class driver defines additional values for the `PathCapability` parameter, the actual values shall be greater than or equal to `IVISWTCH_VAL_CAN_CONNECT_CLASS_EXT_BASE` and less than `IVISWTCH_VAL_CAN_CONNECT_SPECIFIC_EXT_BASE`.

3. If an IVI-COM specific driver implements the `PathCapability` parameter with additional elements in its instrument specific interfaces, the actual values of the additional elements shall be greater than or equal to Can Connect Specific Ext Base.

See Section 9, IviSwtch Function Parameter Value Definitions, for the definitions of Can Connect Specific Ext Base, `IVISWTCH_VAL_CAN_CONNECT_SPECIFIC_EXT_BASE` and `IVISWTCH_VAL_CAN_CONNECT_CLASS_EXT_BASE`.

4.3.2 Connect

Description

This function takes two channel names and, if possible, creates a path between the two channels. If the path already exists, the operation does not count the number of calls. For example, it does not remember that there were two calls to connect, thus requiring two calls to disconnect, but instead returns an error, regardless of whether the order of the two channels is the same or different on the two calls. This is true because paths are assumed to be bi-directional. This class does not handle unidirectional paths. Notice that the IVI spec does not specify the default names for the channels because this depends on the architecture of the switch module. The user can specify aliases for the vendor defined channel names in the IVI Configuration Store.

This function returns as soon as the command is given to the switch module and the switch module is ready for another command. This may be before or after the switches involved settle. Use the Is Debounced function to determine if the switch module has settled. Use the Wait For Debounce function if you want to wait until the switch has debounced.

If an explicit connection already exists between the two specified channels, this function returns the error Explicit Connection Exists without performing any connection operation.

If one of the specified channels is a configuration channel, this function returns the error Is Configuration Channel without performing any connection operation.

If the two specified channels are both connected to a different source, this function returns the error Attempt To Connect Sources without performing any connection operation.

If the two specified channels are the same, this function returns the error Cannot Connect To Itself without performing any connection operation.

If a path cannot be found between the two specified channels, this function returns the error Path Not Found without performing any connection operation.

.NET Prototype

```
void Path.Connect(String channel1,
                  String channel2);
```

COM Prototype

```
HRESULT Path.Connect([in] BSTR Channel1,
                     [in] BSTR Channel2);
```

C Prototype

```
ViStatus IviSwtch_Connect (ViSession Vi, ViConstString Channel1,
                           ViConstString Channel2);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession
Channel1	A string indicating one of the channels of the path.	ViConstString

Channel2	A string indicating one of the channels of the path.	ViConstString

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Explicit Connection Exists	Error: An explicit connection between the channels already exists.
Is Configuration Channel	Error: An explicit connection to a configuration channel is not allowed.
Attempt To Connect Sources	Error: A connection between two different sources is not allowed.
Cannot Connect To Itself	Error: A channel cannot be connected to itself.
Path Not Found	Error: No path was found between the two channels.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
ExplicitConnectionExistsException	An explicit connection between the channels already exists.
IsConfigurationChannelException	An explicit connection to a configuration channel is not allowed.
AttemptToConnectSourcesException	A connection between two different sources is not allowed.
CannotConnectToItselfException	A channel cannot be connected to itself.
PathNotFoundException	No path was found between the two channels.

4.3.3 Disconnect

Description

This function takes two channel names and, if possible, destroys the path between the two channels. The order of the two channels in the operation does not need to be the same as the connect operation. Notice that the IVI specification does not specify what the default names are for the channels as this depends on the architecture of the switch module. The user can specify aliases for the vendor defined channel names in the IVI Configuration Store.

This function returns as soon as the command is given to the switch module and the switch module is ready for another command. This may be before or after the switches involved settle. Use the Is Debounced attribute to see if the switch has settled. Use the Wait For Debounce function if you want to wait until the switch has debounced.

If some connections remain after disconnecting the two specified channels, this function returns the warning Path Remains.

If no explicit path exists between the two specified channels, this function returns the error No Such Path without performing any disconnection operation.

.NET Prototype

```
void Path.Disconnect(String channel1,
                     String channel2);
```

COM Prototype

```
HRESULT Path.Disconnect([in] BSTR Channel1,
                        [in] BSTR Channel2);
```

C Prototype

```
ViStatus IviSwtch_Disconnect (ViSession Vi, ViConstString Channel1,
                              ViConstString Channel2);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession
Channel1	A string indicating one of the channels of the path.	ViConstString
Channel2	A string indicating one of the channels of the path.	ViConstString

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Path Remains	Warning: Some connections remain after disconnecting.
No Such Path	Error: No explicit path exists between the channels.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
NoSuchPathException	No explicit path exists between the channels.

The table below specifies additional class-defined warning events for this method.

Warning	Description
Path Remains	Some connections remain after disconnecting.

4.3.4 Disconnect All

Description

The purpose of this function is to allow the user to disconnect all paths created since Initialize or Reset have been called. This can be used as the test program goes from one sub-test to another to ensure there are no side effects in the switch module.

Notice that some switch modules may not be able to disconnect all paths (such as a scanner that must keep at least one path). In these cases, this function returns the warning Path Remains.

.NET Prototype

```
void Path.DisconnectAll();
```

COM Prototype

```
HRESULT Path.DisconnectAll();
```

C Prototype

```
ViStatus IviSwtch_DisconnectAll (ViSession Vi);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Path Remains	Warning: The instrument is not capable of removing all paths and at least one has been left remaining. Which path remains is vendor specific.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined warning events for this method.

Warning	Description
Path Remains	The instrument is not capable of removing all paths and at least one has been left remaining. Which path remains is vendor specific.

4.3.5 Get Channel Name (IVI-C only)

Description

This function returns the physical name identifier defined by the specific driver for the Channel that corresponds to the one-based index that the user specifies. If the driver defines a qualified channel name, this property returns the qualified name. If the value that the user passes for the `Index` parameter is less than one or greater than the value of the Channel Count attribute, the function returns an empty string in the `Name` parameter and returns an error.

.NET Prototype

`N/A`
(Use the `Channel Name` property)

COM Prototype

`N/A`
(use the `Channel Name` property)

C Prototype

```
ViStatus IviSwtch_GetChannelName (ViSession Vi,
                                  ViInt32 Index,
                                  ViInt32 NameBufferSize,
                                  ViChar Name[]);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Index`	A one-based index that defines which name to return.	`ViInt32`
`Name BufferSize`	The number of bytes in the `ViChar` array that the user specifies for the `Name` parameter.	`ViInt32`

Outputs	Description	Base Type
`Name`	A user-allocated (for IVI-C) or driver-allocated (for IVI-COM) buffer into which the driver stores the channel name The caller may pass `VI_NULL` for this parameter if the `NameBufferSize` parameter is 0.	`ViChar[]`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.6 Get Path

Description

This function returns a list of channels (see the Set Path function for a description on the syntax of path list) that have been connected in order to create the path between the specified channels. The names of the switches as well as the internal configuration of the switch module are vendor specific. This function can be used to return the list of the switches in order to better understand the signal characteristics of the path and to provide the path list for the Set Path function.

The first and last names in the list are the channel names of the path. All channels other than the first and the last channel in the path list are configuration channels. No other channel can be used to generate the path between the two channels.

The only valid paths that can be returned are ones that have been explicitly set via Connect and Set Path functions.

If no explicit path exists between the two specified channels, this function returns the error No Such Path.

.NET Prototype

```
String[] Path.GetPath(String channel1,
                      String channel2);
```

COM Prototype

```
HRESULT Path.GetPath([in] BSTR Channel1,
                     [in] BSTR Channel2,
                     [out, retval] BSTR *PathList);
```

C Prototype

```
ViStatus IviSwtch_GetPath (ViSession Vi, ViConstString Channel1,
                           ViConstString Channel2, ViInt32 PathListBufferSize,
                           ViChar PathList[]);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession
Channel1	A string indicating one of the channels of the path.	ViConstString
Channel2	A string indicating one of the channels of the path.	ViConstString
PathListBufferSize	The number of bytes in the ViChar array that the user specifies for the PathList parameter.	ViInt32

Outputs	Description	Data Type
PathList (C/COM)	A user-allocated (for IVI-C) or driver-allocated (for IVI-COM) buffer into which the driver stores the list of configuration channels used to create a path between the two channels. The caller may pass VI_NULL for this parameter if the PathListBufferSize parameter is 0.	ViChar[]
Return Value (.NET)	A driver-allocated array into which the driver stores the list of configuration channels used to create a path between the two channels.	ViConstString[]

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
No Such Path	Error: No explicit path exists between the channels.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
NoSuchPathException	No explicit path exists between the channels.

4.3.7 Is Debounced (IVI-C only)

Description

The purpose of this function is to inform the user that all the signals flowing through the switch have settled and that it is safe to make a measurement at this time.

.NET Method Prototype

`N/A`
(use the `Path.IsDebounced` property)

COM Method Prototype

`N/A`
(use the `Path.IsDebounced` property)

C Prototype

```
ViStatus IviSwtch_IsDebounced (ViSession Vi, ViBoolean *IsDebounced);
```

Parameters

Inputs	Description	Data Type
`Vi`	Instrument handle	`ViSession`

Outputs	Description	Data Type
`IsDebounced`	Indicates whether the switch has debounced.	`ViBoolean`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.8 Set Path

Description

The IVI Switch is designed to provide automatic routing from channel to channel. However, due to such issues as calibration, it may be necessary to have deterministic control over the path that is created between two channels. This function allows the user to specify the exact path, in terms of the configuration channels used, to create. Notice that the end channel names are the first and last entries in the Path List parameter.

The driver makes a connection between the channels using the configuration channels. The intermediary steps are called legs of the path.

The path list syntax for C and COM is a comma-separated list of path legs. The format of the leg of the path is `ch1->conf1`, where the `ch1` and `conf1` are the two channels the driver used to establish the connection between the first and the last channel. C and COM path lists obey the following rules:

- The second channel of a leg in the path list must be the same as the first channel in the subsequent leg.
- Every channel in the path list other than the first and the last must be a configuration channel.
- Driver channel strings as well as virtual channel names may be used to describe a path leg in a path list.

An example of creating a C or COM path list is:

```
pathList = "ch1->conf1,conf1->ch2";
```

The path list syntax for .NET is a string array of channels. .NET path lists obey the following rules:

- In the array, elements n and n+1 create a path leg.
- Every channel in the path list other than the first and the last must be a configuration channel.
- Driver channel strings as well as virtual channel names may be used to describe a path leg in a path list.

An example of creating a .NET path list is:

```
String[] pathList = {"ch1", "conf1", "ch2"};
```

It should be noticed that, even if users utilize virtual channel names, pathList is not interchangeable since the names of switches within the switch module are not required to be interchangeable and depend on the internal architecture of the switch module. However, it is possible to use the Connect and then Get Path functions to retrieve an already existing path. This allows the user to guarantee that the routing can be recreated exactly.

If the instrument cannot parse a C or COM input path list, this function returns the error Invalid Switch Path without performing any connection operation. Since .NET path lists do not require parsing, this error should not be returned by the .NET method.

If the specified path list is empty, this function returns the error Empty Switch Path without performing any connection operation.

If one of the channels in the path list is a configuration channel that is currently in use, this function returns the error Resource In Use without performing any connection operation.

If an explicit connection is made to a configuration channel, this function returns the error Is Configuration Channel without performing any connection operation.

If one of the non-terminal channels in the path list is not a configuration channel, this function returns the error Not A Configuration Channel without performing any connection operation.

If the path list attempts to connect between two different source channels, this function returns the error Attempt To Connect Sources without performing any connection operation.

If the path list attempts to connect between channels that already have an explicit connection, this function returns the error Explicit Connection Exists without performing any connection operation.

For C and COM path lists, if a leg in the path list does not begin with a channel name, this function returns the error Leg Missing First Channel without performing any connection operation.

For C and COM path lists, if a leg in the path list is missing the second channel, this function returns the error Leg Missing Second Channel without performing any connection operation.

If the first and the second channels in the leg are the same, this function returns the error Channel Duplicated In Leg without performing any connection operation.

If a channel name is duplicated in the path string, this function returns the error Channel Duplicated In Path without performing any connection operation.

For C and COM path lists, if the first channel of a leg in the path is not the same as the second channel in the previous leg, this function returns the error Discontinuous Path without performing any connection operation.

If the path list contains a leg with two channels that cannot be directly connected, this function returns the error Cannot Connect Directly without performing any connection operation.

If a leg in the path contains two channels that are already directly connected, this function returns the error Channels Already Connected without performing any connection operation.

.NET Prototype

```
void Path.SetPath(String[] path);
```

COM Prototype

```
HRESULT Path.SetPath([in] BSTR PathList);
```

C Prototype

```
ViStatus IviSwtch_SetPath (ViSession Vi, ViConstString PathList);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
PathList (C/COM)	List of comma separated channel pairs indicating the path.	ViConstString
pathList (.NET)	Array of channels indicating the path.	ViConstString[]

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Invalid Switch Path	Error: Invalid path list string.
Empty Switch Path	Error: The specified path list string is empty.
Resource In Use	Error: One of the channels in the path is a configuration channel that is in use.
Is Configuration Channel	Error: An explicit connection to a configuration channel is not allowed.
Not A Configuration Channel	Error: One of the non-terminal channels in the path is not a configuration channel.
Attempt To Connect Sources	Error: A connection between two different sources is not allowed.
Explicit Connection Exists	Error: An explicit connection between the channels already exists.
Leg Missing First Channel	Error: A leg in the path does not begin with a channel name.
Leg Missing Second Channel	Error: A leg in the path is missing the second channel.
Channel Duplicated In Leg	Error: The first and the second channels in the leg are the same.
Channel Duplicated In Path	Error: A channel name is duplicated in the path string.
Discontinuous Path	Error: The first channel of a leg in the path is not the same as the second channel in the previous leg.
Cannot Connect Directly	Error: The path contains a leg with two channels that cannot be directly connected.
Channels Already Connected	Error: A leg in the path contains two channels that are already directly connected.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
EmptySwitchPathException	The specified path list string is empty.
ResourceInUseException	One of the channels in the path is a configuration channel that is in use.
IsConfigurationChannelException	An explicit connection to a configuration channel is not allowed.
NotAConfigurationChannelException	One of the non-terminal channels in the path is not a configuration channel.
AttemptToConnectSourcesException	A connection between two different sources is not allowed.

ExplicitConnectionExistsException	An explicit connection between the channels already exists.
ChannelDuplicatedInLegException	The first and the second channels in the leg are the same.
ChannelDuplicatedInPathException	A channel name is duplicated in the path string.
CannotConnectDirectlyException	The path contains a leg with two channels that cannot be directly connected.
ChannelsAlreadyConnectedException	A leg in the path contains two channels that are already directly connected.

4.3.9 Wait For Debounce

Description

The purpose of this function is to wait until the path through the switch is stable (debounced).

If the signals did not settle within the time period the user specified with the `MaxTimeMilliseconds` (C/COM) or `maximumTime` (.NET) parameter, the function returns the Max Time Exceeded error.

.NET Prototype

```
void Path.WaitForDebounce(PrecisionTimeSpan maximumTime);
```

COM Prototype

```
HRESULT Path.WaitForDebounce([in] LONG MaxTimeMilliseconds);
```

C Prototype

```
ViStatus IviSwtch_WaitForDebounce (ViSession Vi, ViInt32 MaxTimeMilliseconds);
```

Parameters

Inputs	Description	Data Type
`Vi`	Instrument handle	`ViSession`
`MaxTimeMilliseconds`	Maximum time (in milliseconds).	`ViInt32`
`maximumTime`	Maximum time.	`PrecisionTimeSpan`

Defined Values for the maximumTime Parameter (.NET)

Name	***Description***		
		Language	***Identifier***
Zero	The function returns immediately without waiting for the debounce to complete.		
		.NET	PrecisionTimeSpan.Zero
Infinite	The function waits indefinitely for the debounce to complete.		
		.NET	PrecisionTimeSpan.MaxValue

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Max Time Exceeded	Error: Maximum time exceeded before the operation completed.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

4.4 IviSwtchBase Behavior Model

The user can access any of the functions in this capability group at anytime. If the user executes the Wait For Debounce function, the driver will block any further operation until the function completes (i.e. all the signals flowing through the switch have settled).

4.5 IviSwtchBase Compliance Notes

1. The driver developer may wish to implement the Settling Time attribute as user readable and write-able, instead of read-only as defined in the attribute specification. This allows the user to specify an arbitrary settling time, which may be shorter than the minimum settling time required by the instrument. Therefore, if a specific driver implements the Settling Time attribute as both user readable and write-able, then the specific driver shall also implement a minimum settling time that is acceptable to the instrument. Any user specified settling time that is shorter than the defined minimum shall be coerced to the minimum settling time.

5 IviSwtchScanner Extension Group

5.1 IviSwtchScanner Overview

The IviSwtchScanner Extension Group defines a set of attributes and functions to perform scanning operations.

5.2 IviSwtchScanner Attributes

The IviSwtchScanner capability group defines the following attributes:

- Continuous Scan
- Is Scanning
- Number of Columns
- Number of Rows
- Scan Advanced Output
- Scan List
- Scan Mode
- Scan Delay
- Trigger Input

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 7, *Attribute ID Definitions*.

5.2.1 Continuous Scan

Data Type	Access	Applies to	Coercion	High Level Functions
ViBoolean	R/W	N/A	N/A	Set Continuous Scan

.NET Property Name

```
Scan.Continuous
```

COM Property Name

```
Scan.Continuous
```

C Constant Name

```
IVISWTCH_ATTR_CONTINUOUS_SCAN
```

Description

If True, the switch module should scan continuously through the scan list. If False, the switch module should scan only once through the scan list.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.2 Is Scanning

Data Type	Access	Applies to	Coercion	High Level Functions
ViBoolean	RO	N/A	N/A	Is Scanning

.NET Property Name

```
Scan.IsScanning
```

COM Property Name

```
Scan.IsScanning
```

C Constant Name

```
IVISWTCH_ATTR_IS_SCANNING
```

Description

If True, the switch module is currently scanning through the scan list (i.e. it is not in the *Idle* state). If False, the switch module is not currently scanning through the scan list (i.e. it is in the *Idle* state).

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.3 Number of Columns

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	RO	N/A	N/A	None

.NET Property Name

```
Scan.NumberOfColumns
```

COM Property Name

```
Scan.NumberOfColumns
```

C Constant Name

```
IVISWTCH_ATTR_NUM_OF_COLUMNS
```

Description

The maximum number of channels on the row of a matrix or scanner. If the switch module is a scanner, this value is the number of input channels. Notice that the number returned is dependent on the Wire Mode attribute.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.4 Number of Rows

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	RO	N/A	N/A	None

.NET Property Name

```
Scan.NumberOfRows
```

COM Property Name

```
Scan.NumberOfRows
```

C Constant Name

```
IVISWTCH_ATTR_NUM_OF_ROWS
```

Description

The maximum number of channels on the column of a matrix or scanner. If the switch module is a scanner, this value is the number of output channels (commons) of the scanner. Notice that the number returned is dependent on the Wire Mode attribute.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.5 Scan Advanced Output

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32 (C/COM)	R/W	N/A	None	Configure Scan Trigger
ViString (.NET)	R/W	N/A	None	Configure Scan Trigger

.NET Property Name

```
Scan.ScannerAdvancedOutput
```

COM Property Name

```
Scan.AdvancedOutput
```

COM Enumeration Name

```
IviSwtchAdvancedOutputEnum
```

C Constant Name

```
IVISWTCH_ATTR_SCAN_ADVANCED_OUTPUT
```

Description

Indicates where the scan advanced output trigger is routed. This trigger is asserted each time a path is created. This trigger shall not be asserted until after sufficient settling time has been given for the path.

If the switch module is currently scanning through the scan list, setting this attribute returns the error Scan In Progress.

Defined Values

In IVI.NET the advanced output trigger is a string. If an IVI driver supports an advanced output trigger and the advanced output trigger is listed in IVI-3.3 *Cross Class Capabilities Specification*, Section 3 then the IVI driver shall accept the standard string for that advanced output trigger. This attribute is case insensitive, but case preserving. That is the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new advanced output trigger strings for triggers that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

Name	***Description***		
		Language	***Identifier***
None	No scan advanced output trigger is sent out of the switch module.		
		C	IVISWTCH_VAL_NONE
		COM	IviSwtchAdvancedOutputNone
GPIB SRQ	The scan advanced output trigger is represented as a GPIB SRQ event.		
		C	IVISWTCH_VAL_GPIB_SRQ

Name	Description		
		Language	Identifier
		COM	IviSwtchAdvancedOutputGPIBSRQ
External	Means the trigger is going out to an external device through a trigger output connection.		
		C	IVISWTCH_VAL_EXTERNAL
		COM	IviSwtchAdvancedOutputExternal
TTL0	The switch asserts TTL0 each time a path is created.		
		C	IVISWTCH_VAL_TTL0
		COM	IviSwtchAdvancedOutputTTL0
TTL1	The switch asserts TTL1 each time a path is created.		
		C	IVISWTCH_VAL_TTL1
		COM	IviSwtchAdvancedOutputTTL1
TTL2	The switch asserts TTL2 each time a path is created.		
		C	IVISWTCH_VAL_TTL2
		COM	IviSwtchAdvancedOutputTTL2
TTL3	The switch asserts TTL3 each time a path is created.		
		C	IVISWTCH_VAL_TTL3
		COM	IviSwtchAdvancedOutputTTL3
TTL4	The switch asserts TTL4 each time a path is created.		
		C	IVISWTCH_VAL_TTL4
		COM	IviSwtchAdvancedOutputTTL4
TTL5	The switch asserts TTL5 each time a path is created.		
		C	IVISWTCH_VAL_TTL5
		COM	IviSwtchAdvancedOutputTTL5
TTL6	The switch asserts TTL6 each time a path is created.		
		C	IVISWTCH_VAL_TTL6
		COM	IviSwtchAdvancedOutputTTL6
TTL7	The switch asserts TTL7 each time a path is created.		
		C	IVISWTCH_VAL_TTL7
		COM	IviSwtchAdvancedOutputTTL7
ECL0	The switch asserts ECL0 each time a path is created.		
		C	IVISWTCH_VAL_ECL0
		COM	IviSwtchAdvancedOutputECL0
ECL1	The switch asserts ECL1 each time a path is created.		
		C	IVISWTCH_VAL_ECL1
		COM	IviSwtchAdvancedOutputECL1
PXI Star	The switch asserts PXI Star each time a path is created.		
		C	IVISWTCH_VAL_PXI_STAR

Name	Description		
		Language	Identifier
		COM	IviSwtchAdvancedOutputPXIStar
RTSI 0	The switch asserts RTSI0 each time a path is created.		
		C	IVISWTCH_VAL_RTSI_0
		COM	IviSwtchAdvancedOutputRTSI0
RTSI 1	The switch asserts RTSI1 each time a path is created.		
		C	IVISWTCH_VAL_RTSI_1
		COM	IviSwtchAdvancedOutputRTSI1
RTSI 2	The switch asserts RTSI2 each time a path is created.		
		C	IVISWTCH_VAL_RTSI_2
		COM	IviSwtchAdvancedOutputRTSI2
RTSI 3	The switch asserts RTSI3 each time a path is created.		
		C	IVISWTCH_VAL_RTSI_3
		COM	IviSwtchAdvancedOutputRTSI3
RTSI 4	The switch asserts RTSI4 each time a path is created.		
		C	IVISWTCH_VAL_RTSI_4
		COM	IviSwtchAdvancedOutputRTSI4
RTSI 5	The switch asserts RTSI5 each time a path is created.		
		C	IVISWTCH_VAL_RTSI_5
		COM	IviSwtchAdvancedOutputRTSI5
RTSI 6	The switch asserts RTSI6 each time a path is created.		
		C	IVISWTCH_VAL_RTSI_6
		COM	IviSwtchAdvancedOutputRTSI6

Compliance Notes

1. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_SPECIFIC_EXT_BASE.

2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_CLASS_EXT_BASE and less than IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_SPECIFIC_EXT_BASE.

3. If an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, the actual values of the additional elements shall be greater than or equal to Scan Advanced Output Class Ext Base.

See Section 8, IviSwtch Attribute Value Definitions, for the definitions of Scan Advanced Output Class Ext Base, IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_SPECIFIC_EXT_BASE and IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_CLASS_EXT_BASE.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.6 Scan Delay

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64 (C/COM)	R/W	N/A	None	Configure Scan Trigger
PrecisionTimeSpan (.NET)	R/W	N/A	None	Configure Scan Trigger

.NET Property Name

```
Scan.Delay
```

COM Property Name

```
Scan.Delay
```

C Constant Name

```
IVISWTCH_ATTR_SCAN_DELAY
```

Description

Specifies the *minimum* length of time from when the path is created to when the scan advanced output trigger is asserted. Due to the design of the switch module, the actual time may be longer. For example, setting a delay of 0 for a switch module that has a fixed debounce delay results in a time of the fixed debounce delay circuit.

Note: For C and COM, the unit for Scan Delay is milliseconds, not seconds.

For .NET, the units are implicit in the definition of Precision Time Span.

If the switch module is currently scanning through the scan list, setting this attribute returns the error Scan In Progress.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.7 Scan List

Data Type	Access	Applies to	Coercion	High Level Functions
ViString	R/W	N/A	None	Configure Scan List

.NET Property Name

```
Scan.List
```

COM Property Name

```
Scan.List
```

C Constant Name

```
IVISWTCH_ATTR_SCAN_LIST
```

Description

The first step in scanning is to tell the driver what channels to scan and in what order. This attribute allows the user to specify the channel list and order by providing a *scan list-string*, which is then parsed by the driver. The basic unit in the scan-list string is the channel pair, which can be separated by special symbols defined in the following table:

The string form of class compliant scan lists may be described in extended Backus-Naur form as follows:

<list> ::= [<triggers>] <pair> [<sequence operator> <pair>]* [<triggers>]
<sequence operator> ::= “&” | <triggers>
<triggers> ::= “;” [“;”]*
<pair> ::= <connect pair> | <disconnect pair>
<disconnect pair> ::= “~” <connect pair>
<connect pair> ::= <channel-name> “->” <channel-name>
<channel-name> ::= A legal channel repeated capability instance name, including qualified names.

The “Connect” pair implicitly breaks previous connections if Scan Mode is Make Before Break of Break After Make. If Scan Mode is None, only pairs with explicit disconnects in the list are opened.

Note the following about the above grammar:

1. It allows waiting for multiple triggers between connecting or disconnecting two channels.
2. It allows for starting the scan by waiting for one or more triggers, and ending the scan by waiting for one or more triggers.

Symbol	Symbol Name	Syntax Example	Description
->	Channel Pair (dash followed by a ‘>’ sign)	CH1->CH2	This symbol signifies a channel pair, which instructs the driver to create a path between the two channels separated by the symbol. In the example, the driver notifies the switch module to create a path between channels CH1 and CH2.

Symbol	Symbol Name	Syntax Example	Description
;	Wait-For-Trigger (semi-colon)	`CH1->CH2 ;` `CH3->CH4`	This character instructs the driver to wait for an input trigger event before proceeding to the next instruction in the scan list string. In the example, the driver notifies the switch module to create a path between channels CH1 and CH2, wait for a trigger, and then create a path between channels CH3 and CH4.

&	List (ampersand)	`CH1->CH2 &`
		`CH3->CH4 ;`
		`A->B`
			This character instructs the driver to connect all the paths separated by the symbol at the same time, before the next trigger event. However, the driver does not guarantee the order of connection, except that all connections are settled before the next trigger event. In the example, the driver notifies the switch module to create a path between channels CH1 and CH2 and between channels CH3 and CH4, not necessarily in that order. The switch module then waits for a trigger before connecting channel A to channel B.

~	Break Connection (tilde)	`~CH1->CH2`	This character instructs the driver to disconnect a path. In the example, the driver notifies the switch module to disconnect channel CH1 from channel CH2. Notice that only path connection events generate scan-advanced triggers. Disconnecting a path will not generate a scan-advanced trigger.

If the switch module is currently scanning through the scan list, setting this attribute returns the error Scan In Progress.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

The table below specifies additional class-defined exceptions for this property.

Exception Class	Description
EmptyScanListException	The given scan list string is empty.
ScanInProgressException	The switch module is currently scanning through the scan list.
InvalidScanListException	The given scan list string does not have the correct syntax, or the syntax cannot be implemented by the switch.

Compliance Issues

When implementing class-compliant methods and properties that set scan lists, IVI specific drivers shall validate that the scan lists conform to the Backus-Naur grammar described above.

5.2.8 Scan Mode

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Scan List

.NET Property Name

Scan.Mode

.NET Enumeration Name

Ivi.Swtch.ScanMode

COM Property Name

Scan.Mode

COM Enumeration Name

IviSwtchScanModeEnum

C Constant Name

IVISWTCH_ATTR_SCAN_MODE

Description

This attribute indicates whether, during a scan, the connections made in the previous connect pair should be broken, and if so, how they should be broken.

If the Scan Mode is None, only channel pairs with explicit disconnect pairs in the scan list are opened.

The idea behind Break Before Make and Break After Make is to ensure that a set of signals being multiplexed down to a single line do or do not short together during a change of channel, typically during a scan (although any switch module can use this feature).

There are specific switches that claim Break Before Make or Break After Make support. This is a special feature of the switch and does not have any impact on the other switches on the module. Therefore, the definition for IVI Switches is that Break Before Make and Break After Make are between channels on a given module, regardless of whether they share a switch or not.

If the switch module is currently scanning through the scan list, setting this attribute returns the error Scan In Progress.

Defined Values

Name	***Description***	
	Language	***Identifier***
Break Before Make	Tells the card to break the previous paths before making the new paths.	

Name	Description		
		Language	Identifier
		.NET	ScanMode.BreakBeforeMake
		C	IVISWTCH_VAL_BREAK_BEFORE_MAKE
		COM	IviSwtchScanModeBreakBeforeMake
Break After Make	Tells the driver to make new paths before breaking the previous paths.		
		.NET	ScanMode.BreakAfterMake
		C	IVISWTCH_VAL_BREAK_AFTER_MAKE
		COM	IviSwtchScanModeBreakAfterMake
None	Indicates that no action should be taken on the previous paths.		
		.NET	ScanMode.None
		C	IVISWTCH_VAL_NONE
		COM	IviSwtchScanModeNone

Compliance Notes

1. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISWTCH_VAL_SCAN_MODE_SPECIFIC_EXT_BASE.

2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISWTCH_VAL_SCAN_MODE_CLASS_EXT_BASE and less than IVISWTCH_VAL_SCAN_MODE_SPECIFIC_EXT_BASE.

3. If an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, the actual values of the additional elements shall be greater than or equal to Scan Mode Specific Ext Base.

See Section 8, IviSwtch Attribute Value Definitions, for the definitions of Scan Mode Specific Ext Base, IVISWTCH_VAL_SCAN_MODE_SPECIFIC_EXT_BASE and IVISWTCH_VAL_SCAN_MODE_CLASS_EXT_BASE.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.9 Trigger Input

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32 (C/COM)	R/W	N/A	None	Configure Scan Trigger
String (.NET)	R/W	N/A	None	Configure Scan Trigger

.NET Property Name

```
Scan.Input
```

COM Property Name

```
Scan.Input
```

COM Enumeration Name

```
IviSwtchTriggerInputEnum
```

C Constant Name

```
IVISWTCH_ATTR_TRIGGER_INPUT
```

Description

Indicates the source of the trigger input. This trigger tells the switch module to advance to the next entry in the scan list and close the specified channel.

If the switch module is currently scanning through the scan list, setting this attribute returns the error Scan In Progress.

Defined Values

In IVI.NET the trigger input is a string. If an IVI driver supports a trigger input and the trigger input is listed in IVI-3.3 *Cross Class Capabilities Specification*, Section 3 then the IVI driver shall accept the standard string for that trigger input. This attribute is case insensitive, but case preserving. That is the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new trigger input strings for trigger inputs that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

Name	***Description***		
		Language	***Identifier***
Immediate	Indicates that the switch module does not wait for a trigger before starting the next entry in the scan list. This is typically done for switch modules that support the Scan Delay attribute and can therefore have the switch module pace itself.		
		C	IVISWTCH_VAL_IMMEDIATE
		COM	IviSwtchTriggerInputImmediate

Name	Description		
		Language	Identifier
Software Trigger	The switch exits the Wait-For-Trigger state when the Send Software Trigger function executes. Refer to the Standardized Cross Class Capabilities specification for a complete description of this value and the Send Software Trigger function		
		C	IVISWTCH_VAL_SOFTWARE_TRIG
		COM	IviSwtchTriggerInputSwTrigFunc
External	Means the trigger is coming from an external source through a trigger input connection.		
		C	IVISWTCH_VAL_EXTERNAL
		COM	IviSwtchTriggerInputExternal
TTL0	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL0.		
		C	IVISWTCH_VAL_TTL0
		COM	IviSwtchTriggerInputTTL0
TTL1	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL1.		
		C	IVISWTCH_VAL_TTL1
		COM	IviSwtchTriggerInputTTL1
TTL2	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL2.		
		C	IVISWTCH_VAL_TTL2
		COM	IviSwtchTriggerInputTTL2
TTL3	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL3.		
		C	IVISWTCH_VAL_TTL3
		COM	IviSwtchTriggerInputTTL3
TTL4	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL4.		
		C	IVISWTCH_VAL_TTL4
		COM	IviSwtchTriggerInputTTL4
TTL5	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL5.		
		C	IVISWTCH_VAL_TTL5
		COM	IviSwtchTriggerInputTTL5
TTL6	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL6.		
		C	IVISWTCH_VAL_TTL6
		COM	IviSwtchTriggerInputTTL6
TTL7	The switch exits the Wait-For-Trigger state when it receives a trigger on TTL7.		

Name	Description		
		Language	Identifier
		C	IVISWTCH_VAL_TTL7
		COM	IviSwtchTriggerInputTTL7
ECL0	The switch exits the Wait-For-Trigger state when it receives a trigger on ECL0.		
		C	IVISWTCH_VAL_ECL0
		COM	IviSwtchTriggerInputECL0
ECL1	The switch exits the Wait-For-Trigger state when it receives a trigger on ECL1.		
		C	IVISWTCH_VAL_ECL1
		COM	IviSwtchTriggerInputECL1
PXI Star	The switch exits the Wait-For-Trigger state when it receives a trigger on PXI Star trigger bus.		
		C	IVISWTCH_VAL_PXI_STAR
		COM	IviSwtchTriggerInputPXIStar
RTSI 0	The switch exits the Wait-For-Trigger state when it receives a trigger on RTSI0.		
		C	IVISWTCH_VAL_RTSI_0
		COM	IviSwtchTriggerInputRTSI0
RTSI 1	The switch exits the Wait-For-Trigger state when it receives a trigger on RTSI1.		
		C	IVISWTCH_VAL_RTSI_1
		COM	IviSwtchTriggerInputRTSI1
RTSI 2	The switch exits the Wait-For-Trigger state when it receives a trigger on RTSI2.		
		C	IVISWTCH_VAL_RTSI_2
		COM	IviSwtchTriggerInputRTSI2
RTSI 3	The switch exits the Wait-For-Trigger state when it receives a trigger on RTSI3.		
		C	IVISWTCH_VAL_RTSI_3
		COM	IviSwtchTriggerInputRTSI3
RTSI 4	The switch exits the Wait-For-Trigger state when it receives a trigger on RTSI4.		
		C	IVISWTCH_VAL_RTSI_4
		COM	IviSwtchTriggerInputRTSI4
RTSI 5	The switch exits the Wait-For-Trigger state when it receives a trigger on RTSI5.		
		C	IVISWTCH_VAL_RTSI_5
		COM	IviSwtchTriggerInputRTSI5

Name	Description		
		Language	Identifier
RTSI 6	The switch exits the Wait-For-Trigger state when it receives a trigger on RTSI6.		
		C	IVISWTCH_VAL_RTSI_6
		COM	IviSwtchTriggerInputRTSI6

Compliance Notes

1. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISWTCH_VAL_TRIGGER_INPUT_SPECIFIC_EXT_BASE.

2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISWTCH_VAL_TRIGGER_INPUT_CLASS_EXT_BASE and less than IVISWTCH_VAL_TRIGGER_INPUT_SPECIFIC_EXT_BASE.

3. If an IVI-COM specific driver implements this attribute with additional elements in its instrument specific interfaces, the actual values of the additional elements shall be greater than or equal to Trigger Input Specific Ext Base.

4. If a specific driver implements any of the defined values in the following table, it shall also implement the corresponding capability group:

Value	Required Capability Group
Software Trigger	IviSwtchSoftwareTrigger

See Section 8, IviSwtch Attribute Value Definitions, for the definitions of Trigger Input Specific Ext Base, IVISWTCH_VAL_TRIGGER_INPUT_SPECIFIC_EXT_BASE and IVISWTCH_VAL_TRIGGER_INPUT_CLASS_EXT_BASE.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.3 IviSwtchScanner Functions

The IviSwtchScanner capability group defines the following functions:

- Abort Scan
- Configure Scan List
- Configure Scan Trigger
- Initiate Scan
- Is Scanning (IVI-C only)
- Set Continuous Scan (IVI-C only)
- Wait For Scan Complete

This section describes the behavior and requirements of each function.

5.3.1 Abort Scan

Description

This function stops the scan begun with Initiate Scan function and returns the switch to the *Idle* state. To determine the status of the scan, call the Is Scanning function. Notice that this operation does not reset the switch module or in any way initialize the state of the switch module. The switch module is simply desensitized from triggers and moved to the *Idle* state.

If the switch module is not currently scanning through the scan list, this function returns the error No Scan In Progress.

.NET Prototype

```
void Scan.Abort();
```

COM Prototype

```
HRESULT Scan.Abort();
```

C Prototype

```
ViStatus IviSwtch_AbortScan (ViSession Vi);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
No Scan In Progress	Error: The switch module is not currently scanning through the scan list.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
NoScanInProgressException	The switch module is not currently scanning through the scan list.

5.3.2 Configure Scan List

Description

Pass the scan list you want the instrument to use. The driver uses this value to set the Scan List attribute.

- The scan list is a string that specifies channel connections and trigger conditions for scanning. After you call the Initiate Scan function, the instrument makes or breaks connections and waits for triggers according to the instructions in the scan list.
- The scan list is comprised of channel names that you separate with special characters. These special characters determine the operation the scanner performs on the channels when it executes this scan list. See Section 0, Scan List for more information about the format of the scan list string.

If the switch module is currently scanning through the scan list, this function returns the error Scan In Progress without configuring the scan list.

If the given scan list string contains incorrect syntax, this function returns the error Invalid Scan List.

If the given scan list string is empty, this function returns the error Empty Scan List.

.NET Prototype

```
void Scan.ConfigureList(String list,
                        Ivi.Swtch.ScanMode mode);
```

COM Prototype

```
HRESULT Scan.ConfigureList([in] BSTR List,
                           [in] IviSwtchScanModeEnum Mode);
```

C Prototype

```
ViStatus IviSwtch_ConfigureScanList (ViSession vi, ViConstString List,
                                     ViInt32 Mode);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession
List	Scan list string. The driver uses this value to set the Scan List attribute. See the attribute description for more details.	ViConstsString
Mode	Scanning mode. The driver uses this value to set the Scan Mode attribute. See the attribute description for more details.	ViInt32

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Empty Scan List	Error: The given scan list string is empty.
Scan In Progress	Error: The switch module is currently scanning through the scan list.
Invalid Scan List	Error: The given scan list string does not have the correct syntax.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
EmptyScanListException	The given scan list string is empty.
ScanInProgressException	The switch module is currently scanning through the scan list.
InvalidScanListException	The given scan list string does not have the correct syntax, or the syntax cannot be implemented by the switch.

5.3.3 Configure Scan Trigger

Purpose

This function configures the scan trigger for the scan list you establish with the Configure Scan List function.

If the switch module is currently scanning through the scan list, this function returns the error Scan In Progress without configuring the scan trigger.

.NET Prototype

```
void Scan.ConfigureTrigger(PrecisionTimeSpan scanDelay,
                           String triggerInput,
                           String scannerAdvanceOutput)
```

COM Prototype

```
HRESULT Scan.ConfigureTrigger([in] DOUBLE ScanDelay,
                              [in] IviSwtchTriggerInputEnum TriggerInput,
                              [in] IviSwtchAdvancedOutputEnum AdvancedOutput)
```

C Prototype

```
ViStatus IviSwtch_ConfigureScanTrigger (ViSession Vi, ViReal64 ScanDelay,
                                        ViInt32 TriggerInput,
                                        ViInt32 AdvancedOutput);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession
ScanDelay	The minimum length of time you want the instrument to wait from the time the instrument creates a path until it asserts a trigger on the Scan Advanced output line (in seconds). The driver uses this value to set the Scan Delay attribute. See the attribute description for more details.	ViReal64 (C/COM) PrecisionTimeSpan (.NET)
TriggerInput	Trigger input. The driver uses this value to set the Trigger Input attribute. See the attribute description for more details.	ViInt32 (C/COM) ViString (.NET)
AdvancedOutput (C/COM) scannerAdvanceOutput (.NET	Scan advanced output. The driver uses this value to set the Scan Advanced Output attribute. See the attribute description for more details.	ViInt32 (C/COM) ViString (.NET)

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Scan In Progress	Error: The switch module is currently scanning through the scan list.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
ScanInProgressException	The switch module is currently scanning through the scan list.

5.3.4 Initiate Scan

Description

This function initiates the scan with the scan list set in the Scan List attribute. If the attribute does not contain a scan list, this function returns the error Empty Scan List. The function is defined to return once the scan has begun. To stop the scanning operation, call Abort Scan.

The first scan advanced output trigger is generated after the Initiate Scan operation, and not when the Scan List attribute is set. If the switch module activates the first switch upon the download of the scan list, the instrument must ensure that no scan advanced output trigger is generated.

Notice that once the switch module is scanning, operations other than reading attributes, Send Software Trigger and Abort Scan are invalid. If any other operation is called on the switch module, that operation shall return the error Scan In Progress.

.NET Prototype

```
void Scan.Initiate();
```

COM Prototype

```
HRESULT Scan.Initiate();
```

C Prototype

```
ViStatus IviSwtch_InitiateScan (ViSession Vi);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Scan In Progress	Error: The switch module is currently scanning through the scan list.
Empty Scan List	Error: No scan list specified.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
ScanInProgressException	The switch module is currently scanning through the scan list.

EmptyScanListException	No scan list specified.

5.3.5 Is Scanning (IVI-C only)

Description

Indicates the state of the switch module. The driver returns the value of the Is Scanning attribute. The value `VI_TRUE` indicates that the switch module is scanning through the scan list. The value `VI_FALSE` indicates that the switch module is idle.

.NET Method Prototype

`N/A`
(use the `Scan.IsScanning` property)

COM Method Prototype

`N/A`
(use the `Scan.IsScanning` property)

C Prototype

```
ViStatus IviSwtch_IsScanning (ViSession Vi, ViBoolean* IsScanning);
```

Parameters

Inputs	Description	Data Type
`Vi`	Instrument handle	`ViSession`

Outputs	Description	Data Type
`IsScanning`	Indicates whether the switch is scanning. The driver returns the value from the Is Scanning attribute. See the attribute description for more details.	`ViBoolean`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

5.3.6 Set Continuous Scan (IVI-C only)

Description

Sets the continuous scan attribute. The driver sets the Continuous Scan attribute. The value `VI_TRUE` indicates that the switch module should continuously scan through the scan list. The value `VI_FALSE` indicates that the switch module should scan only once through the scan list.

.NET Method Prototype

`N/A`
(use the `Scan.Continuous` property)

COM Method Prototype

`N/A`
(use the `Scan.Continuous` property)

C Prototype

```
ViStatus IviSwtch_SetContinuousScan (ViSession Vi, ViBoolean Status);
```

Parameters

Inputs	Description	Data Type
`Vi`	Instrument handle	`ViSession`
`Status`	Continuous scan status. The driver uses this value to set the Continuous Scan attribute. See the attribute description for more details.	`ViBoolean`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

5.3.7 Wait For Scan Complete

Description

This function waits until the instrument stops scanning through the scan list. You specify the maximum length of time for this function to wait until the instrument stops scanning.

If the time you specify elapses before it stops scanning, this function returns a Max Time Exceeded error.

If the switch module is not currently scanning through the scan list, this function returns the error No Scan In Progress.

.NET Prototype

```
void Scan.WaitForScanComplete(PrecisionTimeSpan maximumTime);
```

COM Prototype

```
HRESULT Scan.WaitForScanComplete([in] LONG MaxTimeMilliseconds);
```

C Prototype

```
ViStatus IviSwtch_WaitForScanComplete (ViSession vi, ViInt32
                            MaxTimeMilliseconds);
```

Parameters

Inputs	Description	Data Type
`Vi`	Instrument handle	`ViSession`
`MaxTimeMilliseconds`	Maximum time (ms)	`ViInt32`
`maximumTime`	Maximum time	`PrecisionTimeSpan`

Defined Values for the maximumTime Parameter (.NET)

Name	***Description***		
		Language	***Identifier***
Zero	The function returns immediately without waiting for the scan to complete.		
		.NET	`PrecisionTimeSpan.Zero`
Infinite	The function waits indefinitely for the scan to complete.		
		.NET	`PrecisionTimeSpan.MaxValue`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
No Scan In Progress	Error: The switch module is not currently scanning through the scan list.

Max Time Exceeded	Error: Maximum time exceeded before the operation completed.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below specifies additional class-defined exceptions for this method.

Exception Class	Description
NoScanInProgressException	The switch module is not currently scanning through the scan list

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

5.4 IviSwtchScanner Behavior Model

It is the IVI driver's responsibility to ensure that when the scanning begins a trigger is sent from the switch module if the switch module is configured to assert a trigger on path creation (the Scan Advanced Output attribute). This ensures that if the switch module is using handshake lines with a measurement or source device and also using scanning, the sequence is begun with a trigger from the switch module.

When *not* in the *Idle* or *Reset* state, *all* attributes of the IviSwtch class are read only. Similarly, when *not* in the *Idle* or *Reset* state, the only valid operations are reading of attributes, Reset and Abort Scan.

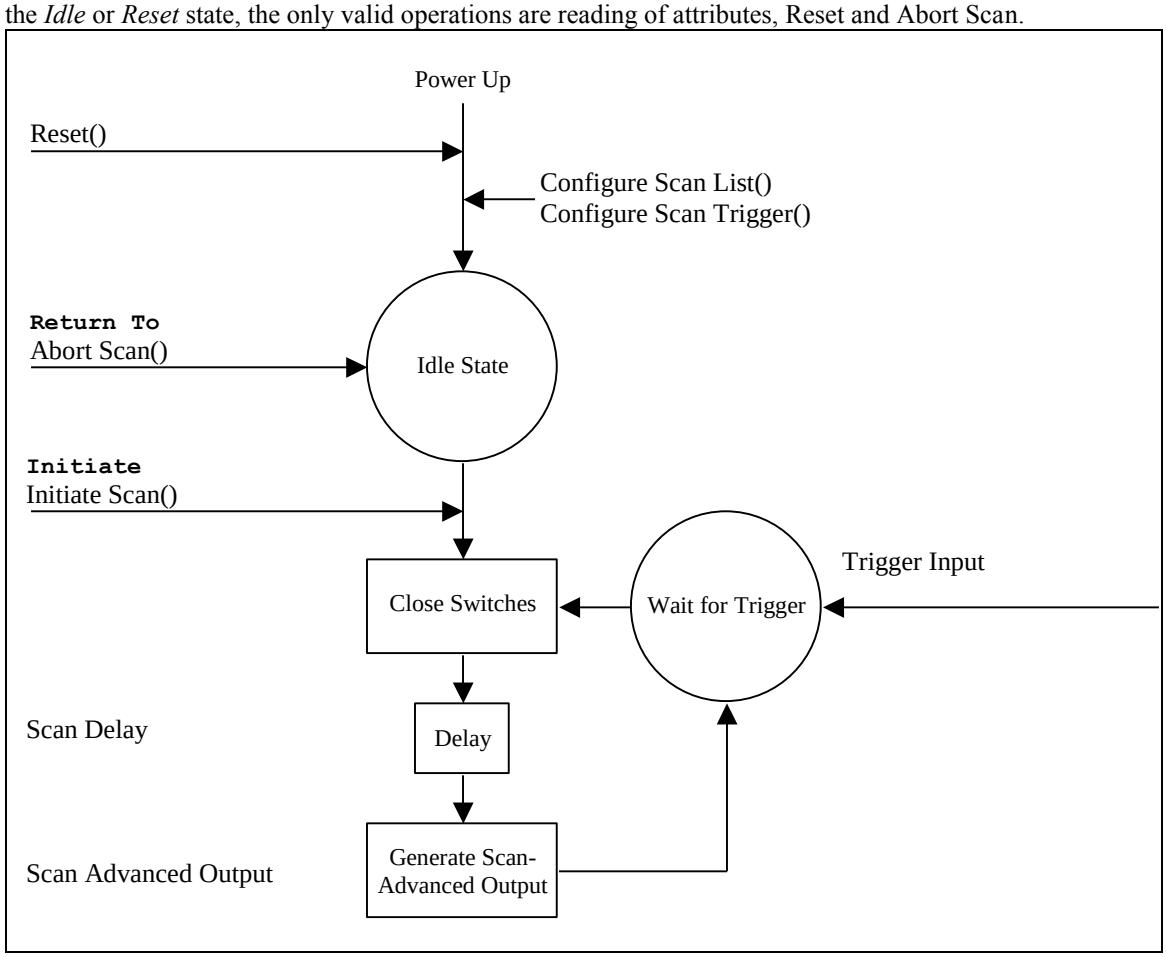


Figure 5-1. IviSwtch Trigger Model

6 IviSwtchSoftwareTrigger Extension Group

6.1 IviSwtchSoftwareTrigger Overview

The IviSwtchSoftwareTrigger Extension Group supports switches that can advance to the next entry in the scan list and close the specified channel based on a software trigger. The user can send a software trigger to cause scan to occur.

6.2 IviSwtchSoftwareTrigger Functions

The IviSwtchSoftwareTrigger extension defines the following functions:

- Send Software Trigger

This section describes the behavior and requirements of this function.

6.2.1 Send Software Trigger

Description

This function sends a software-generated trigger to the instrument. Refer to *IVI-3.3: Standard Cross Class Capabilities Specification* for the complete description of this function.

.NET Prototype

```
void Scan.SendSoftwareTrigger();
```

COM Prototype

```
HRESULT Scan.SendSoftwareTrigger();
```

C Prototype

```
ViStatus IviSwtch_SendSoftwareTrigger (ViSession vi);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional class-defined status codes for this function.

Completion Codes	Description
Trigger Not Software	The trigger input is not set to software trigger.

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET TriggerNotSoftwareException is defined in *IVI-3.2: Inherent Capabilities Specification*.

6.3 IviSwtchSoftwareTrigger Behavior Model

The IviSwtchSoftwareTrigger extension group follows the behavior model of the IviSwtchScanner group. If the Trigger Input attribute is set to Software Trigger, the switch exits the wait-for-trigger state only after the Send Software Trigger function executes.

6.4 IviSwtchSoftwareTrigger Compliance Notes

1. If an instrument driver implements the IviSwtchSoftwareTrigger Capability Group, it must implement the IviSwtchScanner Capability Group.

2. If an instrument driver implements the IviSwtchSoftwareTrigger Capability Group, it must implement the Software Trigger value for the Trigger Input attribute.

7 IviSwtch Attribute ID Definitions

The following table defines the ID value for all IviSwtch class attributes.

Table 7-1. IviSwtch Attributes ID Values

Attribute Name	ID Definition
IVISWTCH_ATTR_IS_SOURCE_CHANNEL	IVI_CLASS_ATTR_BASE + 1
IVISWTCH_ATTR_IS_DEBOUNCED	IVI_CLASS_ATTR_BASE + 2
IVISWTCH_ATTR_IS_CONFIGURATION_CHANNEL	IVI_CLASS_ATTR_BASE + 3
IVISWTCH_ATTR_SETTLING_TIME	IVI_CLASS_ATTR_BASE + 4
IVISWTCH_ATTR_BANDWIDTH	IVI_CLASS_ATTR_BASE + 5
IVISWTCH_ATTR_MAX_DC_VOLTAGE	IVI_CLASS_ATTR_BASE + 6
IVISWTCH_ATTR_MAX_AC_VOLTAGE	IVI_CLASS_ATTR_BASE + 7
IVISWTCH_ATTR_MAX_SWITCHING_DC_CURRENT	IVI_CLASS_ATTR_BASE + 8
IVISWTCH_ATTR_MAX_SWITCHING_AC_CURRENT	IVI_CLASS_ATTR_BASE + 9
IVISWTCH_ATTR_MAX_CARRY_DC_CURRENT	IVI_CLASS_ATTR_BASE + 10
IVISWTCH_ATTR_MAX_CARRY_AC_CURRENT	IVI_CLASS_ATTR_BASE + 11
IVISWTCH_ATTR_MAX_SWITCHING_DC_POWER	IVI_CLASS_ATTR_BASE + 12
IVISWTCH_ATTR_MAX_SWITCHING_AC_POWER	IVI_CLASS_ATTR_BASE + 13
IVISWTCH_ATTR_MAX_CARRY_DC_POWER	IVI_CLASS_ATTR_BASE + 14
IVISWTCH_ATTR_MAX_CARRY_AC_POWER	IVI_CLASS_ATTR_BASE + 15
IVISWTCH_ATTR_CHARACTERISTIC_IMPEDANCE	IVI_CLASS_ATTR_BASE + 16
IVISWTCH_ATTR_WIRE_MODE	IVI_CLASS_ATTR_BASE + 17
IVISWTCH_ATTR_NUM_OF_ROWS	IVI_CLASS_ATTR_BASE + 18
IVISWTCH_ATTR_NUM_OF_COLUMNS	IVI_CLASS_ATTR_BASE + 19
IVISWTCH_ATTR_SCAN_LIST	IVI_CLASS_ATTR_BASE + 20
IVISWTCH_ATTR_SCAN_MODE	IVI_CLASS_ATTR_BASE + 21
IVISWTCH_ATTR_TRIGGER_INPUT	IVI_CLASS_ATTR_BASE + 22
IVISWTCH_ATTR_SCAN_ADVANCED_OUTPUT	IVI_CLASS_ATTR_BASE + 23
IVISWTCH_ATTR_IS_SCANNING	IVI_CLASS_ATTR_BASE + 24
IVISWTCH_ATTR_SCAN_DELAY	IVI_CLASS_ATTR_BASE + 25
IVISWTCH_ATTR_CONTINUOUS_SCAN	IVI_CLASS_ATTR_BASE + 26
IVISWTCH_ATTR_CHANNEL_COUNT	IVI_INHERENT_ATTR_BASE + 203

8 IviSwtch Attribute Value Definitions

This section specifies the actual value for each defined attribute value.

Scan Mode

Value Name	*Language*	*Identifier*	*Actual Value*
None	.NET	ScanMode.None	0
	C	IVISWTCH_VAL_NONE	0
	COM	IviSwtchScanModeNone	0
Break Before Make	.NET	ScanMode.BreakBeforeMake	1
	C	IVISWTCH_VAL_BREAK_BEFORE_MAKE	1
	COM	IviSwtchScanModeBreakBeforeMake	1
Break After Make	.NET	ScanMode.BreakAfterMake	2
	C	IVISWTCH_VAL_BREAK_AFTER_MAKE	2
	COM	IviSwtchScanModeBreakAfterMake	2
Scan Mode Class Ext Base	C	IVISWTCH_VAL_SCAN_MODE_CLASS_EXT_BASE	500
Scan Mode Specific Ext Base	C	IVISWTCH_VAL_SCAN_MODE_SPECIFIC_EXT_BASE	1000
	COM		1000

Scan Action Type (.NET only)

Value Name	*Language*	*Identifier*	*Actual Value*
Connect Path	.NET	ScanActionType.ConnectPath	0
Disconnect Path	.NET	ScanActionType.DisconnectPath	1
Wait For Trigger	.NET	ScanActionType.WaitForTrigger	2

Trigger Input

Value Name	*Language*	*Identifier*	*Actual Value*
Immediate	C	IVISWTCH_VAL_IMMEDIATE	1
	COM	IviSwtchTriggerInputImmediate	1
External	C	IVISWTCH_VAL_EXTERNAL	2
	COM	IviSwtchTriggerInputExternal	2
Software Trigger	C	IVISWTCH_VAL_SOFTWARE_TRIG	3
	COM	IviSwtchTriggerInputSwTrigFunc	3

Value Name	*Language*	*Identifier*	*Actual Value*
TTL0	C	IVISWTCH_VAL_TTL0	111
	COM	IviSwtchTriggerInputTTL0	111
TTL1	C	IVISWTCH_VAL_TTL1	112
	COM	IviSwtchTriggerInputTTL1	112
TTL2	C	IVISWTCH_VAL_TTL2	113
	COM	IviSwtchTriggerInputTTL2	113
TTL3	C	IVISWTCH_VAL_TTL3	114
	COM	IviSwtchTriggerInputTTL3	114
TTL4	C	IVISWTCH_VAL_TTL4	115
	COM	IviSwtchTriggerInputTTL4	115
TTL5	C	IVISWTCH_VAL_TTL5	116
	COM	IviSwtchTriggerInputTTL5	116
TTL6	C	IVISWTCH_VAL_TTL6	117
	COM	IviSwtchTriggerInputTTL6	117
TTL7	C	IVISWTCH_VAL_TTL7	118
	COM	IviSwtchTriggerInputTTL7	118
ECL0	C	IVISWTCH_VAL_ECL0	119
	COM	IviSwtchTriggerInputECL0	119
ECL1	C	IVISWTCH_VAL_ECL1	120
	COM	IviSwtchTriggerInputECL1	120
PXI Star	C	IVISWTCH_VAL_PXI_STAR	125
	COM	IviSwtchTriggerInputPXIStar	125
RTSI 0	C	IVISWTCH_VAL_RTSI_0	140
	COM	IviSwtchTriggerInputRTSI0	140
RTSI 1	C	IVISWTCH_VAL_RTSI_1	141
	COM	IviSwtchTriggerInputRTSI1	141
RTSI 2	C	IVISWTCH_VAL_RTSI_2	142
	COM	IviSwtchTriggerInputRTSI2	142
RTSI 3	C	IVISWTCH_VAL_RTSI_3	143
	COM	IviSwtchTriggerInputRTSI3	143
RTSI 4	C	IVISWTCH_VAL_RTSI_4	144
	COM	IviSwtchTriggerInputRTSI4	144
RTSI 5	C	IVISWTCH_VAL_RTSI_5	145
	COM	IviSwtchTriggerInputRTSI5	145
RTSI 6	C	IVISWTCH_VAL_RTSI_6	146
	COM	IviSwtchTriggerInputRTSI6	146

Value Name	*Language*	*Identifier*	*Actual Value*
Trigger Input Class Ext Base	C	IVISWTCH_VAL_TRIGGER_INPUT_CLASS_EXT_BASE	500
Trigger Input Specific Ext Base	C	IVISWTCH_VAL_TRIGGER_INPUT_SPECIFIC_EXT_BASE	1000
	COM		1000

Scan Advanced Output

Value Name	*Language*	*Identifier*	*Actual Value*
None	C	IVISWTCH_VAL_NONE	0
	COM	IviSwtchAdvancedOutputNone	0
GPIB SRQ	C	IVISWTCH_VAL_GPIB_SRQ	5
	COM	IviSwtchAdvancedOutputGPIBSRQ	5
External	C	IVISWTCH_VAL_EXTERNAL	2
	COM	IviSwtchAdvancedOutputExternal	2
TTL0	C	IVISWTCH_VAL_TTL0	111
	COM	IviSwtchAdvancedOutputTTL0	111
TTL1	C	IVISWTCH_VAL_TTL1	112
	COM	IviSwtchAdvancedOutputTTL1	112
TTL2	C	IVISWTCH_VAL_TTL2	113
	COM	IviSwtchAdvancedOutputTTL2	113
TTL3	C	IVISWTCH_VAL_TTL3	114
	COM	IviSwtchAdvancedOutputTTL3	114
TTL4	C	IVISWTCH_VAL_TTL4	115
	COM	IviSwtchAdvancedOutputTTL4	115
TTL5	C	IVISWTCH_VAL_TTL5	116
	COM	IviSwtchAdvancedOutputTTL5	116
TTL6	C	IVISWTCH_VAL_TTL6	117
	COM	IviSwtchAdvancedOutputTTL6	117
TTL7	C	IVISWTCH_VAL_TTL7	118
	COM	IviSwtchAdvancedOutputTTL7	118
ECL0	C	IVISWTCH_VAL_ECL0	119
	COM	IviSwtchAdvancedOutputECL0	119
ECL1	C	IVISWTCH_VAL_ECL1	120
	COM	IviSwtchAdvancedOutputECL1	120
PXI Star	C	IVISWTCH_VAL_PXI_STAR	125
	COM	IviSwtchAdvancedOutputPXIStar	125

Value Name	*Language*	*Identifier*	*Actual Value*
RTSI 0	C	IVISWTCH_VAL_RTSI_0	140
	COM	IviSwtchAdvancedOutputRTSI0	140
RTSI 1	C	IVISWTCH_VAL_RTSI_1	141
	COM	IviSwtchAdvancedOutputRTSI1	141
RTSI 2	C	IVISWTCH_VAL_RTSI_2	142
	COM	IviSwtchAdvancedOutputRTSI2	142
RTSI 3	C	IVISWTCH_VAL_RTSI_3	143
	COM	IviSwtchAdvancedOutputRTSI3	143
RTSI 4	C	IVISWTCH_VAL_RTSI_4	144
	COM	IviSwtchAdvancedOutputRTSI4	144
RTSI 5	C	IVISWTCH_VAL_RTSI_5	145
	COM	IviSwtchAdvancedOutputRTSI5	145
RTSI 6	C	IVISWTCH_VAL_RTSI_6	146
	COM	IviSwtchAdvancedOutputRTSI6	146
Scan Advanced Output Class Ext Base	C	IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_CLASS_EXT_BASE	500
Scan Advanced Output Specific Ext Base	C	IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_SPECIFIC_EXT_BASE	1000
	COM		1000

8.1 IviSwtch Obsolete Attribute Value Names

The following attribute value names are reserved by the IviSwtch specification 1.0. Future versions of this specification cannot use these names:

- IVISWTCH_VAL_1_WIRE
- IVISWTCH_VAL_2_WIRE
- IVISWTCH_VAL_3_WIRE
- IVISWTCH_VAL_4_WIRE
- IVISWTCH_VAL_GPIB_GET
- IVISWTCH_VAL_SW_TRIG_FUNC

9 IviSwtch Function Parameter Value Definitions

This section specifies the actual values for each function parameter that defines values.

Can Connect

Parameter: `pathCapability`

Value Name	*Language*	*Identifier*	*Actual Value*
Path Available	.NET	Path.Available	0
	C	IVISWTCH_VAL_PATH_AVAILABLE	1
	COM	IviSwtchPathAvailable	1
Path Exists	.NET	Path.Exists	1
	C	IVISWTCH_VAL_PATH_EXISTS	2
	COM	IviSwtchPathExists	2
Path Unsupported	.NET	Path.Unsupported	2
	C	IVISWTCH_VAL_PATH_UNSUPPORTED	3
	COM	IviSwtchPathUnsupported	3
Resource In Use	.NET	Path.ResourceInUse	3
	C	IVISWTCH_VAL_RSRC_IN_USE	4
	COM	IviSwtchPathRsrcInUse	4
Source Conflict	.NET	Path.SourceConflict	4
	C	IVISWTCH_VAL_SOURCE_CONFLICT	5
	COM	IviSwtchPathSourceConflict	5
Channel Not Available	.NET	Path.ChannelNotAvailable	5
	C	IVISWTCH_VAL_CHANNEL_NOT_AVAILABLE	6
	COM	IviSwtchPathChannelNotAvailable	6
Can Connect Class Ext Base	C	IVISWTCH_VAL_CAN_CONNECT_CLASS_EXT_BASE	500
	COM		
Can Connect Specific Ext Base	C	IVISWTCH_VAL_CAN_CONNECT_SPECIFIC_EXT_BASE	1000
	COM		

10 IviSwtch Error and Completion Code Value Definitions

The table below specifies the actual value for each status code that the IviSwtch class specification defines.

Table 10-1. IviSwtch Error and Completion Codes

Error Name	*Description*			
		API	*Identifier*	*Value(hex)*
Path Remains	Some connections remain after disconnecting.			
		.NET		2733A6B6-13E2-4480-9D60-B97FC11B68FC
		C	IVISWTCH_WARN_PATH_REMAINS	0x3FFA2001
		COM	S_IVISWTCH_PATH_REMAINS	0x00042001
Implicit Connection Exists	The implicit connection exists between the channels.			
		.NET		C18A9B2D-C352-4331-A8B5-79BC532923CE
		C	IVISWTCH_WARN_IMPLICIT_CONNECTION_EXISTS	0x3FFA2002
		COM	S_IVISWTCH_IMPLICIT_CONNECTION_EXISTS	0x00042002
Trigger Not Software	The trigger source is not set to software trigger.			
		.NET	Ivi.Driver.TriggerNotSoftwareException	IVI Defined Exception (See IVI-3.2)
		C	IVISWTCH_ERROR_TRIGGER_NOT_SOFTWARE	0xBFFA1001
		COM	E_IVISWTCH_TRIGGER_NOT_SOFTWARE	0x80041001
Invalid Switch Path	Invalid path list string.			
		.NET	InvalidSwitchPathException	N/A
		C	IVISWTCH_ERROR_INVALID_SWITCH_PATH	0xBFFA2001
		COM	E_IVISWTCH_INVALID_SWITCH_PATH	0x80042001
Invalid Scan List	The given scan list string does not have the correct syntax, or the scan list syntax cannot be implemented by the switch.			
		.NET	InvalidScanListException	N/A
		C	IVISWTCH_ERROR_INVALID_SCAN_LIST	0xBFFA2002
		COM	E_IVISWTCH_INVALID_SCAN_LIST	0x80042002
Resource In Use	One of the channels in the path is a configuration channel that is in use.			
		.NET	ResourceInUseException	N/A
		C	IVISWTCH_ERROR_RSRC_IN_USE	0xBFFA2003
		COM	E_IVISWTCH_RSRC_IN_USE	0x80042003

Table 10-1. IviSwtch Error and Completion Codes

Error Name	*Description*			
		API	*Identifier*	*Value(hex)*
Empty Scan List	No scan list specified.			
		.NET	EmptyScanListException	N/A
		C	IVISWTCH_ERROR_EMPTY_SCAN_LIST	0xBFFA2004
		COM	E_IVISWTCH_EMPTY_SCAN_LIST	0x80042004
Empty Switch Path	The specified path list string is empty.			
		.NET	EmptySwitchPathException	N/A
		C	IVISWTCH_ERROR_EMPTY_SWITCH_PATH	0xBFFA2005
		COM	E_IVISWTCH_EMPTY_SWITCH_PATH	0x80042005
Scan In Progress	The switch module is currently scanning through the scan list.			
		.NET	ScanInProgressException	N/A
		C	IVISWTCH_ERROR_SCAN_IN_PROGRESS	0xBFFA2006
		COM	E_IVISWTCH_SCAN_IN_PROGRESS	0x80042006
No Scan In Progress	The switch module is not currently scanning through the scan list.			
		.NET	NoScanInProgressException	N/A
		C	IVISWTCH_ERROR_NO_SCAN_IN_PROGRESS	0xBFFA2007
		COM	E_IVISWTCH_NO_SCAN_IN_PROGRESS	0x80042007
No Such Path	No explicit path exists between the channels.			
		.NET	NoSuchPathException	N/A
		C	IVISWTCH_ERROR_NO_SUCH_PATH	0xBFFA2008
		COM	E_IVISWTCH_NO_SUCH_PATH	0x80042008
Is Configuration Channel	An explicit connection to a configuration channel is not allowed.			
		.NET	IsConfigurationChannelException	N/A
		C	IVISWTCH_ERROR_IS_CONFIGURATION_CHANNEL	0xBFFA2009
		COM	E_IVISWTCH_IS_CONFIGURATION_CHANNEL	0x80042009
Not A Configuration Channel	One of the non-terminal channels in the path is not a configuration channel.			
		.NET	NotAConfigurationChannelException	N/A
		C	IVISWTCH_ERROR_NOT_A_CONFIGURATION_CHANNEL	0xBFFA200A
		COM	E_IVISWTCH_NOT_A_CONFIGURATION_CHANNEL	0x8004200A
Attempt To Connect Sources	A connection between two different sources is not allowed.			
		.NET	AttemptToConnectSourcesException	N/A
		C	IVISWTCH_ERROR_ATTEMPT_TO_CONNECT_SOURCES	0xBFFA200B

Table 10-1. IviSwtch Error and Completion Codes

Error Name	*Description*			
		API	*Identifier*	*Value(hex)*
		COM	E_IVISWTCH_ATTEMPT_TO_CONNECT_SOURCES	0x8004200B
Explicit Connection Exists	An explicit connection between the channels already exists.			
		.NET	ExplicitConnectionExistsException	N/A
		C	IVISWTCH_ERROR_EXPLICIT_CONNECTION_EXISTS	0xBFFA200C
		COM	E_IVISWTCH_EXPLICIT_CONNECTION_EXISTS	0x8004200C
Leg Missing First Channel	A leg in the path does not begin with a channel name.			
		.NET	N/A	N/A
		C	IVISWTCH_ERROR_LEG_MISSING_FIRST_CHANNEL	0xBFFA200D
		COM	E_IVISWTCH_LEG_MISSING_FIRST_CHANNEL	0x8004200D
Leg Missing Second Channel	A leg in the path is missing the second channel.			
		.NET	N/A	N/A
		C	IVISWTCH_ERROR_LEG_MISSING_SECOND_CHANNEL	0xBFFA200E
		COM	E_IVISWTCH_LEG_MISSING_SECOND_CHANNEL	0x8004200E
Channel Duplicated In Leg	The first and the second channels in the leg are the same.			
		.NET	ChannelDuplicatedInLegException	N/A
		C	IVISWTCH_ERROR_CHANNEL_DUPLICATED_IN_LEG	0xBFFA200F
		COM	E_IVISWTCH_CHANNEL_DUPLICATED_IN_LEG	0x8004200F
Channel Duplicated In Path	A channel name is duplicated in the path string.			
		.NET	ChannelDuplicatedInPathException	N/A
		C	IVISWTCH_ERROR_CHANNEL_DUPLICATED_IN_PATH	0xBFFA2010
		COM	E_IVISWTCH_CHANNEL_DUPLICATED_IN_PATH	0x80042010
Path Not Found	No path was found between the two channels.			
		.NET	PathNotFoundException	N/A
		C	IVISWTCH_ERROR_PATH_NOT_FOUND	0xBFFA2011
		COM	E_IVISWTCH_PATH_NOT_FOUND	0x80042011
Discontinuous Path	The first channel of a leg in the path is not the same as the second channel in the previous leg.			
		.NET	N/A	N/A
		C	IVISWTCH_ERROR_DISCONTINUOUS_PATH	0xBFFA2012

Table 10-1. IviSwtch Error and Completion Codes

Error Name	*Description*		
	API	*Identifier*	*Value(hex)*
	COM	E_IVISWTCH_DISCONTINUOUS_PATH	0x80042012
Cannot Connect Directly	The path contains a leg with two channels that cannot be directly connected.		
	.NET	CannotConnectDirectlyException	N/A
	C	IVISWTCH_ERROR_CANNOT_CONNECT_DIRECTLY	0xBFFA2013
	COM	E_IVISWTCH_CANNOT_CONNECT_DIRECTLY	0x80042013
Channels Already Connected	A leg in the path contains two channels that are already directly connected.		
	.NET	ChannelsAlreadyConnectedException	N/A
	C	IVISWTCH_ERROR_CHANNELS_ALREADY_CONNECTED	0xBFFA2014
	COM	E_IVISWTCH_CHANNELS_ALREADY_CONNECTED	0x80042014
Cannot Connect To Itself	A channel cannot be connected to itself.		
	.NET	CannotConnectToItselfException	N/A
	C	IVISWTCH_ERROR_CANNOT_CONNECT_TO_ITSELF	0xBFFA2015
	COM	E_IVISWTCH_CANNOT_CONNECT_TO_ITSELF	0x80042015
Max Time Exceeded	Maximum time exceeded before the operation completed.		
	.NET	Ivi.Driver.MaxTimeExceededException	IVI Defined Exception (See IVI-3.2)
	C	IVISWTCH_ERROR_MAX_TIME_EXCEEDED	0xBFFA2016
	COM	E_IVISWTCH_MAX_TIME_EXCEEDED	0x80042016

Table 10-2 defines the recommended format of the message string associated with the errors. In C, these strings are returned by the Get Error function. In COM, these strings are the description contained in the ErrorInfo object. In .NET these strings are the *Message* property of the exception class thrown by the method or property.

Note: In the description string table entries listed below, **%s** is always used to represent the component name.

Table 10-2. IviSwtch Error Message Strings

Name	Message String
Path Remains	"%s: Some connections remain after disconnecting"
Implicit Connection Exists	"%s: The implicit connection exists between the channels"
Trigger Not Software	"%s: The trigger source is not set to software trigger"
Invalid Switch Path	"%s: Invalid switch path list string"

Table 10-2. IviSwtch Error Message Strings

Name	Message String
Invalid Scan List	"%s: Invalid scan list" "%s: Invalid scan list - the scan list string does not have the correct syntax" "%s: Invalid scan list - the scan list syntax cannot be implemented by the switch."
Resource In Use	"%s: One of the channels in the path is a configuration channel that is in use"
Empty Scan List	"%s: Empty scan list"
Empty Switch Path	"%s: Empty switch path"
Scan In Progress	"%s: Scan in progress"
No Scan In Progress	"%s: No scan in progress"
No Such Path	"%s: No such path"
Is Configuration Channel	"%s: An explicit connection to a configuration channel is not allowed"
Not A Configuration Channel	"%s: One of the non-terminal channels in the path is not a configuration channel"
Attempt To Connect Sources	"%s: Attempt to connect sources"
Explicit Connection Exists	"%s: Explicit connection exists"
Leg Missing First Channel	"%s: Leg missing first channel"
Leg Missing Second Channel	"%s: Leg missing second channel"
Channel Duplicated In Leg	"%s: Channel duplicated in leg"
Channel Duplicated In Path	"%s: Channel duplicated in path"
Path Not Found	"%s: Path not found"
Discontinuous Path	"%s: Discontinuous path"
Cannot Connect Directly	"%s: Cannot connect directly"
Channels Already Connected	"%s: Channels already connected"
Cannot Connect To Itself	"%s: Cannot connect to itself"
Max Time Exceeded	"%s: Max time exceeded"

10.1 IVI.NET IviSwtch Exceptions and Warnings

This section defines the list of IVI.NET exceptions and warnings that are specific to the IviSwtch class. For general information on IVI.NET exceptions and warnings, refer to *IVI-3.1: Driver Architecture Specification* and section 12, *Common IVI.NET Exceptions and Warnings*, of *IVI-3.2: Inherent Capabilities Specification*.

The IVI.NET exceptions defined in this specification are declared in the Ivi.Swtch namespace.

- AttemptToConnectSourcesException
- CannotConnectDirectlyException
- CannotConnectToItselfException
- ChannelDuplicatedInLegException
- ChannelDuplicatedInPathException
- ChannelsAlreadyConnectedException
- EmptyScanListException
- EmptySwitchPathException
- ExplicitConnectionExistsException
- InvalidScanListException
- IsConfigurationChannelException
- NoScanInProgressException
- NoSuchPathException
- NotAConfigurationChannelException
- PathNotFoundException
- ResourceInUseException
- ScanInProgressException

10.1.1 AttemptToConnectSourcesException

Description

This exception is used when an attempt is made to connect two channels that are both sources.

Constructors

```
Ivi.Swtch.AttemptToConnectSourcesException(String channel1Name,
                                           String channel2Name);

Ivi.Swtch.AttemptToConnectSourcesException();

Ivi.Swtch.AttemptToConnectSourcesException(String message);

Ivi.Swtch.AttemptToConnectSourcesException(String message,
                                           System.Exception innerException);
```

Message String

```
A connection between two different sources is not allowed.
Channel 1 Name: <channel1Name>.
Channel 2 Name: <channel2Name>.
```

Parameters

Inputs	Description	Base Type
`channel1Name`	The name of the first channel	`String`
`channel2Name`	The name of the second channel	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.2 CannotConnectDirectlyException

Description

This exception is used when an attempt is made to connect two channels that cannot be directly connected.

Constructors

```
Ivi.Swtch.CannotDirectlyConnectException(String channel1Name,
                                         String channel2Name);

Ivi.Swtch.CannotDirectlyConnectException();

Ivi.Swtch.CannotDirectlyConnectException(String message);

Ivi.Swtch.CannotDirectlyConnectException(String message,
                                         System.Exception innerException);
```

Message String

```
The path contains a leg with two channels that cannot be directly connected.
Channel 1 Name: <channel1Name>.
Channel 2 Name: <channel2Name>.
```

Parameters

Inputs	Description	Base Type
`channel1Name`	The name of the first channel	`String`
`channel2Name`	The name of the second channel	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.3 CannotConnectToItselfException

Description

This exception is used when the driver attempts to connect a specified channel to itself.

Constructors

```
Ivi.Swtch.CannotConnectToItselfException(String message,
                                         String channelName);

Ivi.Swtch.CannotConnectToItselfException();

Ivi.Swtch.CannotConnectToItselfException(String message);

Ivi.Swtch.CannotConnectToItselfException(String message,
                                         System.Exception innerException);
```

Message String

```
A channel cannot be connected to itself.
Channel name: <channelName>
```

Parameters

Inputs	Description	Base Type
`channelName`	The channel name.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.4 ChannelDuplicatedInLegException

Description

This exception is used when the driver detects that two channels in a leg are the same.

Recommended Constructors

```
Ivi.Swtch.ChannelDuplicatedInLegException(String message,
                                          String channelName);

Ivi.Swtch.ChannelDuplicatedInLegException();

Ivi.Swtch.ChannelDuplicatedInLegException(String message);

Ivi.Swtch.ChannelDuplicatedInLegException(String message,
                                          System.Exception innerException);
```

Message String

```
The two channels in the leg are the same.
Channel name: <channelName>
```

Parameters

Inputs	Description	Base Type
`channelName`	The channel name.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.5 ChannelDuplicatedInPathException

Description

This exception is used when the driver detects that a channel name is duplicated in the path.

Constructors

```
Ivi.Swtch.ChannelDuplicatedInPathException(String message,
                                           String channelName);

Ivi.Swtch.ChannelDuplicatedInPathException();

Ivi.Swtch.ChannelDuplicatedInPathException(String message);

Ivi.Swtch.ChannelDuplicatedInPathException(String message,
                                           System.Exception innerException);
```

Message String

```
A channel name is duplicated in the path.
Channel name: <channelName>
```

Parameters

Inputs	Description	Base Type
`channelName`	The channel name.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.6 ChannelsAlreadyConnectedException

Description

This exception is used when an attempt is made to connect two channels that are already directly connected.

Constructors

```
Ivi.Swtch.ChannelsAlreadyConnectedException(String channel1Name,
                                            String channel2Name);

Ivi.Swtch.ChannelsAlreadyConnectedException();

Ivi.Swtch.ChannelsAlreadyConnectedException(String message);

Ivi.Swtch.ChannelsAlreadyConnectedException(String message,
                                            System.Exception innerException);
```

Message String

```
A leg in the path contains two channels that are already directly connected.
Channel 1 Name: <channel1Name>.
Channel 2 Name: <channel2Name>.
```

Parameters

Inputs	Description	Base Type
`Channel1Name`	The name of the first channel	`String`
`channel2Name`	The name of the second channel	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.7 EmptyScanListException

Description

This exception is used when no scan list is specified.

Constructors

```
Ivi.Swtch.EmptyScanListException();

Ivi.Swtch.EmptyScanListException(String message);

Ivi.Swtch.EmptyScanListException(String message,
                                 System.Exception innerException);
```

Message String

```
No scan list is specified.
```

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.8 EmptySwitchPathException

Description

This exception is used when the switch path is empty.

Constructors

```
Ivi.Swtch.EmptySwitchPathException();

Ivi.Swtch.EmptySwitchPathException(String message);

Ivi.Swtch.EmptySwitchPathException(String message,
                                   System.Exception innerException);
```

Message String

```
The switch path is empty.
```

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.9 ExplicitConnectionExistsException

Description

This exception is used when an attempt is made to connect two channels that are already explicitly connected.

Constructors

```
Ivi.Swtch.ExplicitConnectionExistsException(String channel1Name,
                                            String channel2Name);

Ivi.Swtch.ExplicitConnectionExistsException();

Ivi.Swtch.ExplicitConnectionExistsException(String message);

Ivi.Swtch.ExplicitConnectionExistsException(String message,
                                            System.Exception innerException);
```

Message String

```
An explicit connection between the channels already exists.
Channel 1 Name: <channel1Name>.
Channel 2 Name: <channel2Name>.
```

Parameters

Inputs	Description	Base Type
`Channel1Name`	The first channel	`String`
`channel2Name`	The second channel	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.10 InvalidScanListException

Description

This exception is used when the driver finds that the given scan list string does not have the correct syntax, or the scan list syntax cannot be implemented by the switch.

Recommended Constructors

```
Ivi.Swtch.InvalidScanListException(String message,
                                   String scanList);

Ivi.Swtch.InvalidScanListException();

Ivi.Swtch.InvalidScanListException(String message);

Ivi.Swtch.InvalidScanListException(String message,
                                   System.Exception innerException);
```

Message String

```
The given scan list string does not have the correct syntax, or the scan list
syntax cannot be implemented by the switch.
Scan list: <scanList>
```

Parameters

Inputs	Description	Base Type
`scanList`	The scan list string.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.11 IsConfigurationChannelException

Description

This exception is used when the driver detects an attempt to explicitly connect to a configuration channel.

Constructors

```
Ivi.Swtch.IsConfigurationChannelException(String message,
                                          String channelName);

Ivi.Swtch.IsConfigurationChannelException();

Ivi.Swtch.IsConfigurationChannelException(String message);

Ivi.Swtch.IsConfigurationChannelException(String message,
                                          System.Exception innerException);
```

Message String

```
An explicit connection to a configuration channel is not allowed.
Channel name: <channelName>
```

Parameters

Inputs	Description	Base Type
`channelName`	The channel name.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.12 NoScanInProgressException

Description

This exception is used when the driver expects that the switch is currently scanning through the scan list, but it is not.

Constructors

```
Ivi.Swtch.NoScanInProgressException();

Ivi.Swtch.NoScanInProgressException(String message);

Ivi.Swtch.NoScanInProgressException(String message,
                                   System.Exception innerException);
```

Message String

```
The switch is not currently scanning through the scan list.
```

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.13 NoSuchPathException

Description

This exception is used when no explicit path exists between the channels.

Constructors

```
Ivi.Swtch.NoSuchPathException(String channel1Name,
                              String channel2Name);

Ivi.Swtch.NoSuchPathException();

Ivi.Swtch.NoSuchPathException(String message);

Ivi.Swtch.NoSuchPathException(String message,
                              System.Exception innerException);
```

Message String

```
No explicit path exists between the channels.
Channel 1 Name: <channel1Name>.
Channel 2 Name: <channel2Name>.
```

Parameters

Inputs	Description	Base Type
`Channel1Name`	The first channel	`String`
`channel2Name`	The second channel	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.14 NotAConfigurationChannelException

Description

This exception is used when the driver finds that one of the non-terminal channels in the path is not a configuration channel.

Constructors

```
Ivi.Swtch.NotAConfigurationChannelException(String message,
                                            String channelName);

Ivi.Swtch.NotAConfigurationChannelException();

Ivi.Swtch.NotAConfigurationChannelException(String message);

Ivi.Swtch.NotAConfigurationChannelException(String message,
                                            System.Exception innerException);
```

Message String

```
One of the non-terminal channels in the path is not a configuration channel.
Channel name: <channelName>
```

Parameters

Inputs	Description	Base Type
`channelName`	The channel name.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.15 PathNotFoundException

Description

This exception is used when the driver expects to find a path between two channels, but the path is not found.

Constructors

```
Ivi.Swtch.PathNotFoundException(String channel1Name,
                                String channel2Name);

Ivi.Swtch.PathNotFoundException();

Ivi.Swtch.PathNotFoundException(String message);

Ivi.Swtch.PathNotFoundException(String message,
                                System.Exception innerException);
```

Message String

```
No path was found between the channels.
Channel 1 Name: <channel1Name>.
Channel 2 Name: <channel2Name>.
```

Parameters

Inputs	Description	Base Type
`Channel1Name`	The first channel	`String`
`channel2Name`	The second channel	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.16 ResourceInUseException

Description

This exception is used when the driver finds that one of the channels in the path is a configuration channel that is in use.

Constructors

```
Ivi.Swtch.ResourceInUseException(String message,
                                 String channelName);

Ivi.Swtch.ResourceInUseException();

Ivi.Swtch.ResourceInUseException(String message);

Ivi.Swtch.ResourceInUseException(String message,
                                 System.Exception innerException);
```

Message String

```
One of the channels in the path is a configuration channel that is in use.
Channel name: <channelName>
```

Parameters

Inputs	Description	Base Type
`channelName`	The channel name.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

10.1.17 ScanInProgressException

Description

This exception is used when the driver expects that the switch is not currently scanning through the scan list, but it is.

Constructors

```
Ivi.Swtch.ScanInProgressException();

Ivi.Swtch.ScanInProgressException(String message);

Ivi.Swtch.ScanInProgressException(String message,
                                  System.Exception innerException);
```

Message String

```
The switch is currently scanning through the scan list.
```

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

11 IviSwtch Hierarchies

11.1 IviSwtch .NET Hierarchy

The full IviSwtch .NET Hierarchy includes the Inherent Capabilities Hierarchy as defined in Section 4.1, *.NET Inherent Capabilities* of *IVI-3.2: Ineherent Capabilities Specification*. To avoid redundancy, it is omitted here.

Table 11-1. IviSwtch .NET Hierarchy

.NET Interface Hierarchy	Generic Name	Type
Channels		
Count	Channel Count	P
[]	**Channel Item**	
IsConfigurationChannel	Is Configuration Channel	P
IsSourceChannel	Is Source Channel	P
Name	Channel Name	P
Characteristics		
ACCurrentCarryMax	AC Current Carry Max	P
ACCurrentSwitchingMax	AC Current Switching Max	P
ACPowerCarryMax	AC Power Carry Max	P
ACPowerSwitchingMax	AC Power Switching Max	P
ACVoltageMax	AC Voltage Max	P
Bandwidth	Bandwidth	P
DCCurrentCarryMax	DC Current Carry Max	P
DCCurrentSwitchingMax	DC Current Switching Max	P
DCPowerCarryMax	DC Power Carry Max	P
DCPowerSwitchingMax	DC Power Switching Max	P
DCVoltageMax	DC Voltage Max	P
Impedance	Characteristic Impedance	P
SettlingTime	Settling Time	P
WireMode	Wire Mode	P
Path		
CanConnect	Can Connect	M
Connect	Connect	M
Disconnect	Disconnect	M
DisconnectAll	Disconnect All	M
GetPath	Get Path	M
IsDebounced	Is Debounced	P
SetPath	Set Path	M

Table 11-1. IviSwtch .NET Hierarchy

.NET Interface Hierarchy	Generic Name	Type
WaitForDebounce	Wait For Debounce	M
Scan		
Abort	Abort Scan	M
AdvancedOutput	Scan Advanced Output	P
ConfigureList	Configure Scan List	M
ConfigureTrigger	Configure Scan Trigger	M
Continuous	Continuous Scan	P
Delay	Scan Delay	P
Initiate	Initiate Scan	M
Input	Trigger Input	P
IsScanning	Is Scanning	P
List	Scan List	P
Mode	Scan Mode	P
NumberOfColumns	Number of Columns	P
NumberOfRows	Number of Rows	P
SendSoftwareTrigger	Send Software Trigger	M
WaitForScanComplete	Wait For Scan Complete	M

11.1.1 IviSwtch .NET Interfaces

In addition to implementing IVI inherent capabilities interfaces, IIviSwtch interfaces contain interface reference properties for accessing the following IviSwtch interfaces:

1. IIviSwtchPath
2. IIviSwtchScan
3. IIviSwtchChannels

The IIviSwtchChannels interface contains methods and properties for accessing a collection of objects that implement the IIviSwtchChannel interface.

The IIviSwtchChannel interface contains an interface reference property for accessing the IIviSwtchCharacteristics interface.

11.1.2 Interface Reference Properties

Interface reference properties are used to navigate the IviSwtch .NET hierarchy. This section describes the interface reference properties that the IviSwtch, IIviSwtchChannels, and IIviSwtchChannel interfaces define. All interface reference properties are read-only.

Data Type	.NET Property Name
IIviSwtchPath	Path
IIviSwtchScan	Scan
IIviSwtchChannels	Channels

Data Type	.NET Property Name
IIviSwtchChannel	Channels[]
IIviSwtchCharacteristics	Characteristics

11.2 IviSwtch COM Hierarchy

The full IviSwtch COM Hierarchy includes the Inherent Capabilities Hierarchy as defined in Section 4.2, *COM Inherent Capabilities* of *IVI-3.2: Ineherent Capabilities Specification*. To avoid redundancy, it is omitted here.

Table 11-2. IviSwtch COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
Channels		
Count	Channel Count	P
Name	Channel Name	P
Item		
IsConfigurationChannel	Is Configuration Channel	P
IsSourceChannel	Is Source Channel	P
Characteristics		
ACCurrentCarryMax	AC Current Carry Max	P
ACCurrentSwitchingMax	AC Current Switching Max	P
ACPowerCarryMax	AC Power Carry Max	P
ACPowerSwitchingMax	AC Power Switching Max	P
ACVoltageMax	AC Voltage Max	P
Bandwidth	Bandwidth	P
Impedance	Characteristic Impedance	P
DCCurrentCarryMax	DC Current Carry Max	P
DCCurrentSwitchingMax	DC Current Switching Max	P
DCPowerCarryMax	DC Power Carry Max	P
DCPowerSwitchingMax	DC Power Switching Max	P
DCVoltageMax	DC Voltage Max	P
SettlingTime	Settling Time	P
WireMode	Wire Mode	P
Path		
IsDebounced	Is Debounced	P
CanConnect	Can Connect	M
Connect	Connect	M
Disconnect	Disconnect	M
DisconnectAll	Disconnect All	M

Table 11-2. IviSwtch COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
GetPath	Get Path	M
SetPath	Set Path	M
WaitForDebounce	Wait For Debounce	M
Scan		
Continuous	Continuous Scan	P
IsScanning	Is Scanning	P
NumberOfColumns	Number of Columns	P
NumberOfRows	Number of Rows	P
AdvancedOutput	Scan Advanced Output	P
Delay	Scan Delay	P
List	Scan List	P
Mode	Scan Mode	P
Input	Trigger Input	P
Abort	Abort Scan	M
ConfigureList	Configure Scan List	M
ConfigureTrigger	Configure Scan Trigger	M
Initiate	Initiate Scan	M
SendSoftwareTrigger	Send Software Trigger	M
WaitForScanComplete	Wait For Scan Complete	M

11.2.1 IviSwtch COM Interfaces

In addition to implementing IVI inherent capabilities interfaces, IIviSwtch interfaces contain interface reference properties for accessing the following IviSwtch interfaces:

1. IIviSwtchPath
2. IIviSwtchScan
3. IIviSwtchChannels

The IIviSwtchChannels interface contains methods and properties for accessing a collection of objects that implement the IIviSwtchChannel interface.

The IIviSwtchChannel interface contains an interface reference property for accessing the IIviSwtchCharacteristics interface.

Table 11-3. IviSwtch Interface GUIDs lists the interfaces that this specification defines and their GUIDs.

Table 11-3. IviSwtch **Interface GUIDs**

Interface	GUID
IIviSwtch	47ed527e-a398-11d4-ba58-000064657374
IIviSwtchPath	47ed527f-a398-11d4-ba58-000064657374

Table 11-3. IviSwtch **Interface GUIDs**

Interface	GUID
IIviSwtchScan	47ed5280-a398-11d4-ba58-000064657374
IIviSwtchChannels	47ed5281-a398-11d4-ba58-000064657374
IIviSwtchChannel	47ed5282-a398-11d4-ba58-000064657374
IIviSwtchCharacteristics	47ed5283-a398-11d4-ba58-000064657374

11.2.2 Interface Reference Properties

Interface reference properties are used to navigate the IviSwtch .NET hierarchy. This section describes the interface reference properties that the IviSwtch, IIviSwtchChannels, and IIviSwtchChannel interfaces define. All interface reference properties are read-only.

Data Type	Access
IIviSwtchPath*	Path
IIviSwtchScan*	Scan
IIviSwtchChannels*	Channels
IIviSwtchChannel*	Channel
IIviSwtchCharacteristics*	Characteristics

11.2.3 IviSwtch COM Category

The IviSwtch class COM Category shall be “IviSwtch”, and the Category ID (CATID) shall be {47ed5157-a398-11d4-ba58-000064657374}.

11.3 IviSwtch C Function Hierarchy

The IviSwtch class function hierarchy is shown in the following table. The full IviSwtch C Function Hierarchy includes the Inherent Capabilities Hierarchy as defined in Section 4.3, *C Inherent Capabilities* of *IVI-3.2: Inherent Capabilities Specification*. To avoid redundancy, it is omitted here.

☞ **Note:** ***To reduce complexity, the individual Set and Get attribute functions required by IVI are not shown in the following table.***

Name or Class	Function Name
Configuration…	
Configure Scan List	`IviSwtch_ConfigureScanList`
Configure Scan Trigger	`IviSwtch_ConfigureScanTrigger`
Set Continuous Scan	`IviSwtch_SetContinuousScan`
Route…	
Connect Channels	`IviSwtch_Connect`
Disconnect Channels	`IviSwtch_Disconnect`
Disconnect All Channels	`IviSwtch_DisconnectAll`
Switch Is Debounced?	`IviSwtch_IsDebounced`
Wait For Debounce	`IviSwtch_WaitForDebounce`
Can Connect Channels?	`IviSwtch_CanConnect`
Paths…	
Set Path	`IviSwtch_SetPath`
Get Path	`IviSwtch_GetPath`
Scan…	
Initiate Scan	`IviSwtch_InitiateScan`
Abort Scan	`IviSwtch_AbortScan`
Switch Is Scanning?	`IviSwtch_IsScanning`
Wait For Scan To Complete	`IviSwtch_WaitForScanComplete`
Send Software Trigger	`IviSwtch_SendSoftwareTrigger`
Utility…	
Get Channel Name	`IviSwtch_GetChannelName`

11.4 IviSwtch C Attribute Hierarchy

The IviSwtch class attribute hierarchy is shown in the following table. The full IviSwtch C Attribute Hierarchy includes the Inherent Capabilities Hierarchy as defined in Section 4.3, *C Inherent Capabilities* of *IVI-3.2: Inherent Capabilities Specification*. To avoid redundancy, it is omitted here.

Table 11-5. IviSwtch C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
Channel Configuration	
Is Source Channel	`IVISWTCH_ATTR_IS_SOURCE_CHANNEL`
Is Configuration Channel	`IVISWTCH_ATTR_IS_CONFIGURATION_CHANNEL`
Module Characteristics	

Table 11-5. IviSwtch C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
Is Debounced	IVISWTCH_ATTR_IS_DEBOUNCED
Settling Time	IVISWTCH_ATTR_SETTLING_TIME
Bandwidth	IVISWTCH_ATTR_BANDWIDTH
Maximum Carry AC Current	IVISWTCH_ATTR_MAX_CARRY_AC_CURRENT
Maximum Switching AC Current	IVISWTCH_ATTR_MAX_SWITCHING_AC_CURRENT
Maximum Carry AC Power	IVISWTCH_ATTR_MAX_CARRY_AC_POWER
Maximum Switching AC Power	IVISWTCH_ATTR_MAX_SWITCHING_AC_POWER
Maximum AC Voltage	IVISWTCH_ATTR_MAX_AC_VOLTAGE
Maximum Carry DC Current	IVISWTCH_ATTR_MAX_CARRY_DC_CURRENT
Maximum Switching DC Current	IVISWTCH_ATTR_MAX_SWITCHING_DC_CURRENT
Maximum Carry DC Power	IVISWTCH_ATTR_MAX_CARRY_DC_POWER
Maximum Switching DC Power	IVISWTCH_ATTR_MAX_SWITCHING_DC_POWER
Maximum DC Voltage	IVISWTCH_ATTR_MAX_DC_VOLTAGE
Characteristic Impedance	IVISWTCH_ATTR_CHARACTERISTIC_IMPEDANCE
Scanning Configuration	
Scan List	IVISWTCH_ATTR_SCAN_LIST
Scan Mode	IVISWTCH_ATTR_SCAN_MODE
Continuous Scan	IVISWTCH_ATTR_CONTINUOUS_SCAN
Trigger Input	IVISWTCH_ATTR_TRIGGER_INPUT
Scan Advanced Output	IVISWTCH_ATTR_SCAN_ADVANCED_OUTPUT
Is Scanning	IVISWTCH_ATTR_IS_SCANNING
Scan Delay	IVISWTCH_ATTR_SCAN_DELAY
Matrix Configuration	
Number of Columns	IVISWTCH_ATTR_NUM_OF_COLUMNS
Number of Rows	IVISWTCH_ATTR_NUM_OF_ROWS
Wire Mode	IVISWTCH_ATTR_WIRE_MODE

Appendix A. Specific Drivers Development Guidelines

A.1 Introduction

This section describes situations driver developers should be aware of when developing a specific instrument driver that complies with the IviSwtch class.

A.2 Disabling Unused Extensions

Specific drivers are required to disable extension capability groups that an application program does not explicitly use. The specific driver can do so by setting the attributes of an extension capability group to the values that this section recommends. A specific driver can set these values for all extension capability groups when the Initialize, Initialize With Options or Reset functions execute. This assumes that the extension capability groups remain disabled until the application program explicitly uses them. For the large majority of instruments, this assumption is true.

Under certain conditions, a specific driver might have to implement a more complex approach. For some instruments, configuring a capability group might affect instrument settings that correspond to an unused extension capability group. If these instrument settings affect the behavior of the instrument, then this might result in an interchangeability problem. If this can occur, the specific driver must take appropriate action so that the instrument settings that correspond to the unused extension capability group do not affect the behavior of the instrument when the application program performs an operation that might be affected by those settings.

The remainder of this section recommends attribute values that effectively disable each extension capability group.

Disabling the IviSwtchSoftwareTrigger Extension Group

The IviSwtchSoftwareTrigger extension group affects the instrument behavior only when the Trigger Input attribute is set to Software Trigger. Therefore, this specification does not recommend attribute values that disable the IviSwtchSoftwareTrigger extension group.

A.3 Implementing the Analog Bus

Many switch modules have a special output connection known as the analog bus. This connection allows for the chaining of multiple switch modules together. For example, four 1x64 multiplexers can be chained together through the analog bus to create a 1x256 multiplexer. While this can always be done with external wiring, the analog bus typically has special switches that allow the switch modules to connect or disconnect from the analog bus.

If the switch module does have an analog bus, it should be treated in the same way as a normal input or output channel. This means that the connection point and analog bus switch (if implemented) are considered a channel to which you create paths. An example of a multiplexer with analog bus is shown below.

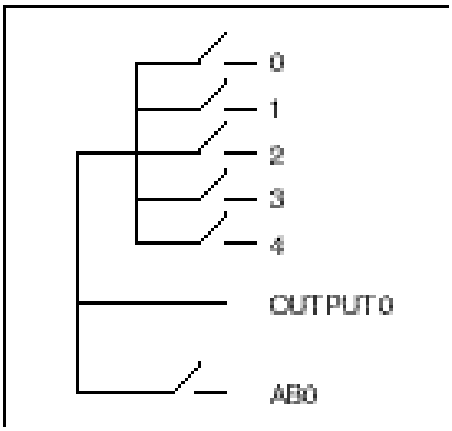


Figure A-1. Analog Bus Example

As you can see from Figure 3, to connect `1` to `AB0` is a matter of calling the Connect function. It is important to note, however, that by doing so you have implicitly created a path to `OUTPUT0`. This is a good example of how the IviSwtch driver is designed to abstract the concepts of the switch module, but not completely remove the requirement that the user understand the architecture of the switch module he or she is using. In this case, there has been no explicit path created from `1` to `OUTPUT0` or from `OUTPUT0` to `AB0`, therefore Disconnect fails if these paths are specified. However, it is the switch driver's responsibility to know about these side effects when dealing with such things as excluding two sources from being connected together or a channel being in use.

A.4 Scanning

The purpose of the scanning functions is to allow the user to achieve high-speed control of the switch module as well as deterministic timing of the measurements. Some switch modules do not have hardware FIFO-based architecture, which means that all scanning is done in software. In these situations the controller must be inserted in the trigger handshake between the measurement device and the switch module. It is the responsibility of the instrument driver provider to clearly document the fact that the module supports only software scanning to insure that the user understands the ability of this switch module and instrument driver. If inserting the controller in the trigger handshake is not possible due to hardware constraints, then the driver should not support the scanning functions and require the user to use the fundamental functions only.

The switch generates the first scan advanced output signal when the Initiate Scan function executes. If the hardware cannot support this functionality, then the driver should not download the scan list until the call to Initiate Scan. The reason for this is so that the measurement device can be configured to take the first measurement on the first scan advanced output trigger.

In **`BREAK BEFORE MAKE`**, any existing paths must be disconnected before performing any scan. At each ";" in the scan list, all of the previously closed connections are opened before proceeding to the next connections in the scan list. IviSwtch requires that any scan list in **`BREAK BEFORE MAKE`** ends with a ";" so that no connection paths remain after a scan completes.

In **`BREAK AFTER MAKE`**, any existing paths must be disconnected after performing the scan. At each ";" in the scan list, all of the previously closed connections are opened after executing to the next connections in the scan list. A switch card that supports **`BREAK AFTER MAKE`** places the card in a safe state when your program calls either Disconnect All or Abort Scan. This guarantees the current continuity for inductive loads.

If the value of the Scan Mode attribute is None, you can start a scan with connection paths already existing on the switch card. Connection commands in the scan list create new connections and leave the existing

paths untouched. This scan mode does not require a ";" at the end of a scan list; in this case, the switch card does not wait for a final trigger before terminating the scan list. When a scan completes, the paths created by the scan remain connected until the application explicitly disconnects them.

During a scan in any of the above scan modes, calling the Abort Scan function causes the scan to stop abruptly. If the driver is able to maintain the knowledge of the established connections during scanning, then the Abort Scan function does not need to perform any further operation. However, if the driver is unable to maintain such knowledge, the recommended behavior is to have the Abort Scan function call the Disconnect All function after aborting the scan.

A.5 Scan Delay

The Scan Delay attribute is specified to provide a clocking mechanism from the switch module. However, most switch modules provide an internal, fixed delay (known as the debounce delay) that is always generated. This guarantees that the path has settled to its new state and the signal is passing through cleanly before the switch module alerts the external device (typically a measurement or source device). Therefore, when a user specifies a time in this attribute that is less then that of the debounce delay, the switch module *must* wait the longer of the two time periods for debounce and settling.

A.6 Multi Switch Module Instrument Drivers

The definition of the IviSwtch class incorporates both simple switch topologies, such as a 1xn multiplexer and a nxm matrix, as well as complex switch topologies, such as multiple switch modules wired together. This means that an IviSwtch instrument driver that operates on smaller IviSwtch instrument drivers is possible. At this level, it is then possible to provide a complete signal routing of a switch system.

However, it might not be possible to create such a generic, high-level instrument driver that supports scanning. The reason for this is that the configuration of switches often need to be changed during a scan when multiple modules are wired together. The IviSwtch instrument driver definition does not provide a way to access these configuration switches in an interoperable fashion. In these cases, an IviSwtch driver built specifically for a grouping of certain switch modules is possible. This higher-level instrument driver is then interoperable, but it is not possible to swap out lower level switch modules without modifying the instrument driver.

A.7 General Purpose Switches

A general-purpose switch is simply a collection of basic switches (Form A, Form C, etc.) that are independent from each other. These switches are then used to perform such actions as controlling power to motors, fans, etc. The IviSwtch class has been designed primarily to handle routing and scanning issues that users face with complex switch systems. However, the IviSwtch class can also handle these general-purpose topologies. To support such a switch module, the input and output of the switch must be independently named so that a path can be created between them.

For example, a Form A switch would have two names, such as `Switch1Input` and `Switch1Output`. Opening and closing this switch is then accomplished by calling Connect and Disconnect.

When developing a driver for form C switches, you may implement the Disconnect call in at least three different ways:

- Disconnecting common (C) from normally open (NO) or normally closed (NC) never performs any action other than marking C as being disconnected from both NO and NC. This approach can save relay life by minimizing the number of relay operations.

- Disconnecting C from NO or NC flips the relay state. For example, if C and NC are connected, a call to disconnect C from NC has the effect of connecting C to NO physically, but the driver considers C as being disconnected from both NO and NC. (To subsequently connect C to NO, the

application developer should make an explicit call to Connect C to NO so that the driver considers C and NO connected.)

- Disconnecting C from NO or NC always brings the relay to the normally closed state. As in the previous case, C is connected to NC only physically. The driver considers C as being disconnected from both NO and NC.

A.8 Wire Mode Attribute

The Wire Mode attribute specifies the number of connections in a channel. In some cases, a channel may be connecting a bus that has a number of conductors. In those cases, the specific driver may create constants that describe the types of bus the switch is capable of connecting, such as `WIRE_MODE_GPIB_DATA` to describe the GPIB data bus. However, in order to achieve maximum interchangeability, a constant should have a value that corresponds to the number of connectors in the bus. For example, the `WIRE_MODE_GPIB_DATA` constant would have a corresponding value of 8.

Appendix B. Interchangeability Checking Rules

B.1 Introduction

IVI drivers have a feature called interchangeability checking. Interchangeability checking returns a warning when it encounters a situation were the application program might not produce the same behavior when the user attempts to use a different instrument.

B.2 When to Perform Interchangeability Checking

Interchangeability checking occurs when all of the following conditions are met:

- The Interchange Check attribute is set to True
- The user has set the value of any of the IviSwtchScanner extension group attributes
- The user calls one of the following functions.
  - Connect
  - Set Path
  - Initiate Scan

B.3 Interchangeability Checking Rules

Interchangeability checking is performed on a capability group basis. When enabled, interchangeability checking is always performed on the base capability group. In addition, interchangeability checking is performed on extension capability groups for which the user has ever set any of the attributes of the group. If the user has never set any attributes of an extension capability group, interchangeability checking is not performed on that group.

In general interchangeability warnings are generated if the following conditions are encountered:

- An attribute that affects the behavior of the instrument is not in a state that the user specifies.
- The user sets a class driver defined attribute to an instrument-specific value.
- The user configures the value of an attribute that the class defines as read-only. In a few cases the class drivers define read-only attributes that specific drivers might implement as read/write.

The remainder of this section defines additional rules and exceptions for each capability group.

IviSwtchBase Capability Group

No additional interchangeability rules or exceptions are defined for the IviSwtchBase capability group.

IviSwtchScanner Capability Group

No additional interchangeability rules or exceptions are defined for the IviSwtchScanner capability group.