



IVI-4.6: IviSwtch Class Specification

April, 2002 Edition
Revision 3.0

Important Information

The IviSwch 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, or contact the IVI Foundation at 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation via email at ivilistserver@ivifoundation.org, via the web site at www.ivifoundation.org, or you can write to the IVI Foundation, 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.

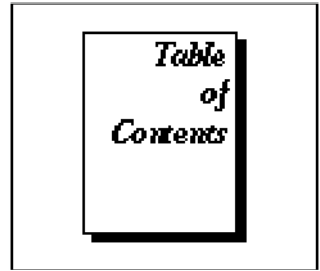
Warranty

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.



1.	Overview of the IviSwtch Specification	8
1.1	Introduction	8
1.2	IviSwtch Class Overview.....	8
1.3	References.....	9
1.4	Definitions of Terms and Acronyms.....	9
2.	IviSwtch Class Capabilities	11
2.1	Introduction	11
2.2	IviSwtch Group Names.....	11
2.3	IviSwtch Repeated Capability Names.....	11
2.3.1	Channel	11
3.	General Requirements	12
3.1	Minimum Class Compliance	12
3.1.1	Disable	12
3.2	Capability Group Compliance.....	12
4.	IviSwtchBase Capability Group	13
4.1	IviSwtchBase Overview.....	13
4.2	IviSwtchBase Attributes	13
4.2.1	AC Current Carry Max	14
4.2.2	AC Current Switching Max	15
4.2.3	AC Power Carry Max.....	16
4.2.4	AC Power Switching Max.....	17
4.2.5	AC Voltage Max	18
4.2.6	Bandwidth.....	19
4.2.7	Channel Count.....	20
4.2.8	Channel Item (COM only)	21
4.2.9	Channel Name (COM only).....	22
4.2.10	Characteristic Impedance.....	23
4.2.11	DC Current Carry Max	24
4.2.12	DC Current Switching Max	25
4.2.13	DC Power Carry Max.....	26
4.2.14	DC Power Switching Max.....	27
4.2.15	DC Voltage Max	28
4.2.16	Is Configuration Channel.....	29
4.2.17	Is Debounced	30
4.2.18	Is Source Channel.....	31
4.2.19	Settling Time	32

4.2.20	Wire Mode	33
4.3	IviSwchBase Functions	34
4.3.1	Can Connect.....	35
4.3.2	Connect	37
4.3.3	Disconnect	39
4.3.4	Disconnect All.....	40
4.3.5	Get Channel Name (IVI-C only)	41
4.3.6	Get Path.....	42
4.3.7	Is Debounced (IVI-C only)	44
4.3.8	Set Path	45
4.3.9	Wait For Debounce	48
4.4	IviSwchBase Behavior Model.....	49
4.5	IviSwchBase Compliance Notes	49
5.	IviSwchScanner Extension Group.....	50
5.1	IviSwchScanner Overview	50
5.2	IviSwchScanner Attributes.....	50
5.2.1	Continuous Scan.....	51
5.2.2	Is Scanning.....	52
5.2.3	Number of Columns.....	53
5.2.4	Number of Rows	54
5.2.5	Scan Advanced Output	55
5.2.6	Scan Delay.....	58
5.2.7	Scan List.....	59
5.2.8	Scan Mode	61
5.2.9	Trigger Input.....	63
5.3	IviSwchScanner Functions	67
5.3.1	Abort Scan	68
5.3.2	Configure Scan List.....	69
5.3.3	Configure Scan Trigger.....	70
5.3.4	Initiate Scan.....	71
5.3.5	Is Scanning (IVI-C only)	72
5.3.6	Set Continuous Scan (IVI-C only)	73
5.3.7	Wait For Scan Complete.....	74
5.4	IviSwchScanner Behavior Model	75
6.	IviSwchSoftwareTrigger Extension Group.....	76
6.1	IviSwchSoftwareTrigger Overview	76
6.2	IviSwchSoftwareTrigger Functions	76
6.2.1	Send Software Trigger	77
6.3	IviSwchSoftwareTrigger Behavior Model	78
6.4	IviSwchSoftwareTrigger Compliance Notes.....	78
7.	IviSwch Attribute ID Definitions.....	79
8.	IviSwch Attribute Value Definitions.....	80
8.1	IviSwch Obsolete Attribute Value Names.....	83

9.	IviSwtch Function Parameter Value Definitions	84
10.	IviSwtch Error and Completion Code Value Definitions	85
11.	IviSwtch Hierarchies.....	90
11.1	IviSwtch COM Hierarchy.....	90
11.1.1	IviSwtch COM Interfaces	92
11.1.2	Interface Reference Properties	93
11.1.3	IviSwtch COM Category.....	95
11.2	IviSwtch C Function Hierarchy.....	96
11.3	IviSwtch C Attribute Hierarchy.....	96
Appendix A.	Specific Drivers Development Guidelines	98
A.1	Introduction	98
A.2	Disabling Unused Extensions.....	98
A.3	Implementing the Analog Bus.....	98
A.4	Scanning.....	99
A.5	Scan Delay.....	100
A.6	Multi Switch Module Instrument Drivers.....	100
A.7	General Purpose Switches	100
A.8	Wire Mode Attribute.....	100
Appendix B.	Interchangeability Checking Rules.....	101
B.1	Introduction	101
B.2	When to Perform Interchangeability Checking.....	101
B.3	Interchangeability Checking Rules	101
Appendix C.	ANSI C Include File.....	102
Appendix D.	COM IDL File.....	106
D.1	IviSwtchTypeLib.idl	106
D.2	IviSwtch.idl.....	107
D.3	IviSwtchEnglish.idl	118

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 0.1	June 2, 1997	Original draft.
Revision 0.2	June 30, 1997	Clarified meaning of string based names and removed functions that referred to channels through numbers.
Revision 0.3	August 7, 1997	Changed the definition of the fundamental class to include general switch routing based on the feedback from the first IVI League Meeting.
Revision 0.4	October 1, 1997	Added the ability to set a specific path for routing in the switch module (IviSwtch_SetPath).
Revision 0.5	October 21, 1997	Modifications from the IVI League Meeting. Removed the BREAK_MODE attribute and changed MULTI_POINT_CAPABLE to SOURCE. Expanded definitions and the number of channel attributes for switch characteristics (such as max. switching current). Also added DisconnectAll function. Finally, removed ROUTE_OUTPUT and the analog bus is treated as another channel.
Revision 0.6	January 24, 1998	Incorporated edits from previous IVI League Meeting. Reformatted spec into the new layout. Added/reviewed text descriptions. Added IviSwtch_SendSoftwareTrigger() operation. Updated scan list syntax.
Revision 1.0a	April 21, 1998	Reformatted the document into yet another new layout. Added attributes and functions, new error and completion codes. Amended Scan list syntax rules.
Revision 1.0b1	June 26, 1998	More edits of text, to fit the format and to clarify unclearly defined terms. Added scanning discussion to specific driver development guidelines.
Revision 1.1	August 21, 1998	Technical Publications review and edit. Changes to template information.

Table 1. IviSwtch Class Specification Revisions

Revision Number	Date of Revision	Revision Notes
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 MaxTimeMilliseconds.
Revision 3.0	April 12, 2002	Released version 3.0, including the COM interface specification. No content change from Voting Candidate 7.

1. Overview of the IviSwch Specification

1.1 Introduction

This specification defines the IVI class for switches. The IviSwch class is designed to support the typical switches as well as common extended functionality found in specialized switch modules. This section summarizes the *IviSwch 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:

- ? IviSwch Class Overview
- ? The definitions of terms and acronyms
- ? References

1.2 IviSwch Class Overview

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

An IviSwch 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 IviSwch class conceptualizes the switch as an instrument that can establish paths between its I/O channels.

The IviSwch 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 IviSwch base capability group is described in Section 4, *IviSwchBase Capability Group*.

In addition to the base capability group, the IviSwch 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 IviSwchScanner extension group.

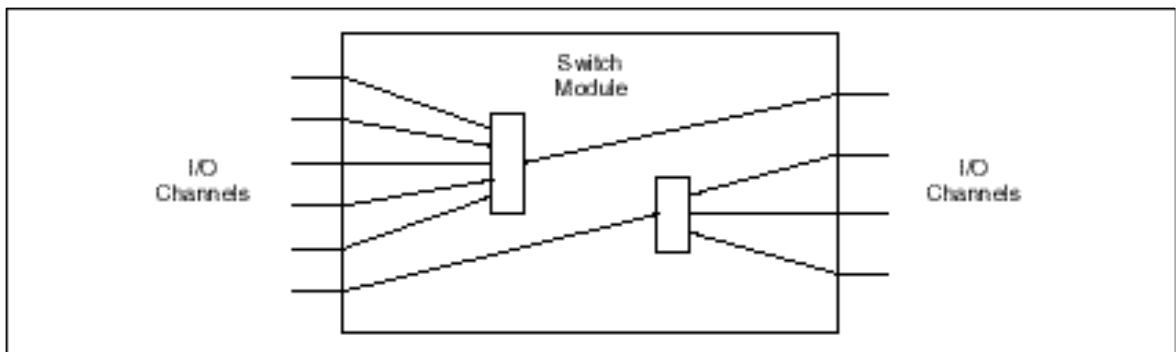


Figure 1-1. Switch Module

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-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*, 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 “Channel”.

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 IviSwch 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 IviSwchBase 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. IviSwchBase Capability Group

4.1 IviSwchBase Overview

The IviSwchBase Capability Group defines attributes and their values to determine the characteristics of I/O channels and the status of paths. The IviSwchBase 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 IviSwchBase Attributes

The IviSwchBase 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 only)
- ? Channel Name (COM 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, *IviSwch Attribute ID Definitions*.

4.2.1 AC Current Carry Max

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

COM Property Name

`Channels.Item().Characteristics.ACCurrentCarryMax`

COM Enumeration Name

N/A

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.

4.2.2 AC Current Switching Max

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

COM Property Name

`Channels.Item().Characteristics.ACCurrentSwitchingMax`

COM Enumeration Name

N/A

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.

4.2.3 AC Power Carry Max

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

COM Property Name

`Channels.Item().Characteristics.ACPowerCarryMax`

COM Enumeration Name

N/A

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.

4.2.4 AC Power Switching Max

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

COM Property Name

`Channels.Item().Characteristics.ACPowerSwitchingMax`

COM Enumeration Name

N/A

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.

4.2.5 AC Voltage Max

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

COM Property Name

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

COM Enumeration Name

N/A

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.

4.2.6 Bandwidth

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

COM Property Name

`Channels.Item().Characteristics.Bandwidth`

COM Enumeration Name

N/A

C Constant Name

`IVISWTCH_ATTR_BANDWIDTH`

Description

The bandwidth, in Hertz, for the channel. Specifies the maximum frequency for the signal you want the instrument to accommodate 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.

4.2.7 Channel Count

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

COM Property Name

`Channels.Count`

COM Enumeration Name

N/A

C Constant Name

`IVISWTCH_ATTR_CHANNEL_COUNT`

Description

Returns the number of available channels.

4.2.8 Channel Item (COM only)

Data Type	Access	Applies to	Coercion	High Level Functions
IIviSwTchChannel*	RO	Channels	N/A	None

COM Property Name

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

COM Enumeration Name

N/A

C Constant Name

N/A

Description

The Channel Item attribute requires a string parameter which is the name of one of the channel. It returns an interface pointer which can be used to control the attributes and other functionality of that channel.

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

Return Values

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.

4.2.9 Channel Name (COM only)

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

COM Property Name

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

COM Enumeration Name

N/A

C Constant Name

N/A

Description

This attribute 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 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.

4.2.10 Characteristic Impedance

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

COM Property Name

`Channels.Item().Characteristics.Impedance`

COM Enumeration Name

N/A

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.

4.2.11 DC Current Carry Max

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

COM Property Name

`Channels.Item().Characteristics.DCCurrentCarryMax`

COM Enumeration Name

N/A

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.

4.2.12 DC Current Switching Max

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

COM Property Name

`Channels.Item().Characteristics.DCCurrentSwitchingMax`

COM Enumeration Name

N/A

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.

4.2.13 DC Power Carry Max

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

COM Property Name

`Channels.Item().Characteristics.DCPowerCarryMax`

COM Enumeration Name

N/A

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.

4.2.14 DC Power Switching Max

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

COM Property Name

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

COM Enumeration Name

N/A

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.

4.2.15 DC Voltage Max

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

COM Property Name

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

COM Enumeration Name

N/A

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.

4.2.16 Is Configuration Channel

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

COM Property Name

`Channels.Item().IsConfigurationChannel`

COM Enumeration Name

N/A

C Constant Name

`IVISWTCH_ATTR_IS_CONFIGURATION_CHANNEL`

Description

Specifies whether the specific driver uses the channel for internal path creation. 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.

Defined Values

<i>Name</i>	<i>Description</i>	
	<i>Language</i>	<i>Identifier</i>
True	The channel is no longer accessible to the user and can be used by the specific driver for path creation.	
	C	VI_TRUE
	COM	VARIANT_TRUE
False	The channel is considered a standard channel and can be explicitly connected to another channel.	
	C	VI_FALSE
	COM	VARIANT_FALSE

4.2.17 Is Debounced

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

COM Property Name

Path.IsDebounced

COM Enumeration Name

N/A

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. It indicates that the signal going through the switch module is valid, assuming that the switches in the path have the correct characteristics.

Defined Values

Name	Description	
	Language	Identifier
True	The switch module has settled from the switching commands and completed the debounce.	
	C	VI_TRUE
	COM	VARIANT_TRUE
False	The switch module has not settled.	
	C	VI_FALSE
	COM	VARIANT_FALSE

4.2.18 Is Source Channel

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

COM Property Name

`Channels.Item().IsSourceChannel`

COM Enumeration Name

N/A

C Constant Name

`IVISWTCH_ATTR_IS_SOURCE_CHANNEL`

Description

Allows the user to declare a particular channel as 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.

Defined Values

Name	Description	
	Language	Identifier
True	The channel is a source channel.	
	C	VI_TRUE
	COM	VARIANT_TRUE
False	The channel is not a source channel.	
	C	VI_FALSE
	COM	VARIANT_FALSE

4.2.19 Settling Time

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

COM Property Name

`Channels.Item().Characteristics.SettlingTime`

COM Enumeration Name

N/A

C Constant Name

`IVISWTCH_ATTR_SETTLING_TIME`

Description

The maximum total settling time, in seconds, 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.

4.2.20 Wire Mode

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

COM Property Name

`Channels.Item().Characteristics.WireMode`

COM Enumeration Name

N/A

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.

4.3 IviSwchBase Functions

The IviSwchBase 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.

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	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.	
	C	IVISWTCH_VAL_PATH_AVAILABLE
	COM	IviSwTchPathAvailable
Path Exists	The explicit path between the channels already exists.	
	C	IVISWTCH_VAL_PATH_EXISTS
	COM	IviSwTchPathExists

Name	Description	
	Language	Identifier
Path Unsupported	The instrument is not capable of creating a path between the two channels.	
	C	IVISWTCH_VAL_PATH_UNSUPPORTED
	COM	IviSwrchPathUnsupported
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.	
	C	IVISWTCH_VAL_RSRC_IN_USE
	COM	IviSwrchPathRsrcInUse
Source Conflict	The instrument cannot create a path between the two channels because both are connected to a different source channel.	
	C	IVISWTCH_VAL_SOURCE_CONFLICT
	COM	IviSwrchPathSourceConflict
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.	
	C	IVISWTCH_VAL_CHANNEL_NOT_AVAILABLE
	COM	IviSwrchPathChannelNotAvailable

Return Values

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.

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, *IviSwrch 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.

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

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.

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.

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

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.

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.

COM Prototype

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

C Prototype

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

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession

Return Values

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.

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 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.

COM Prototype

N/A
(use the `Channels.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 <code>ViChar</code> array that the user specifies for the <code>Name</code> 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 <code>VI_NULL</code> for this parameter if the <code>NameBufferSize</code> parameter is 0.	ViChar[]

Return Values

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 the list of comma separated channel pairs (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.

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	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 Values

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.

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.

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

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. These intermediary steps are called legs of the path. 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. The path list syntax is a comma-separated list of path legs that 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.

An example of a path list is: `ch1->conf1, conf1->ch2`

It should be noticed that this string 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 the input path list string, this function returns the error Invalid Switch Path without performing any connection operation.

If the specified path list string 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.

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.

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.

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.

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	List of comma separated channel pairs indicating the path.	ViConstString

Return Values

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.

Completion Codes	Description
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.

4.3.9 Wait For Debounce

Description

The purpose of this function is to wait until all the signals flowing through the switch have settled.

If the signals did not settle within the time period the user specified with the `MaxTimeMilliseconds` parameter, the function returns the Max Time Exceeded error.

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

Return Values

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.

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

COM Property Name

Scan.Continuous

COM Enumeration Name

N/A

C Constant Name

IVISWTCH_ATTR_CONTINUOUS_SCAN

Description

Indicates whether the switch module should scan continuously through the scan list or only scan once through the scan list.

Defined Values

Name	Description	
	Language	Identifier
True	The switch module should scan continuously through the scan list.	
	C	VI_TRUE
	COM	VARIANT_TRUE
False	The switch module should scan only once through the scan list.	
	C	VI_FALSE
	COM	VARIANT_FALSE

5.2.2 Is Scanning

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

COM Property Name

Scan.IsScanning

COM Enumeration Name

N/A

C Constant Name

IVISWTCH_ATTR_IS_SCANNING

Description

States whether the switch module is currently scanning through the scan list (i.e. it is not in the *Idle* state).

Defined Values

Name	Description	
	Language	Identifier
True	The switch module is currently scanning through the scan list.	
	C	VI_TRUE
	COM	VARIANT_TRUE
False	The switch module is not currently scanning through the scan list.	
	C	VI_FALSE
	COM	VARIANT_FALSE

5.2.3 Number of Columns

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

COM Property Name

Scan.NumberOfColumns

COM Enumeration Name

N/A

C Constant Name

IVISWTCH_ATTR_NUM_OF_COLUMNS

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 input channels. Notice that the number returned is dependent on the Wire Mode attribute.

5.2.4 Number of Rows

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

COM Property Name

Scan.NumberOfRows

COM Enumeration Name

N/A

C Constant Name

IVISWTCH_ATTR_NUM_OF_ROWS

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 output channels (commons) of the scanner. Notice that the number returned is dependent on the Wire Mode attribute.

5.2.5 Scan Advanced Output

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

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

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
	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

Name	Description	
	Language	Identifier
TTL1	The switch asserts TTL1 each time a path is created.	
	C	IVISWTCH_VAL_TTL1
	COM	IviSwrchAdvancedOutputTTL1
TTL2	The switch asserts TTL2 each time a path is created.	
	C	IVISWTCH_VAL_TTL2
	COM	IviSwrchAdvancedOutputTTL2
TTL3	The switch asserts TTL3 each time a path is created.	
	C	IVISWTCH_VAL_TTL3
	COM	IviSwrchAdvancedOutputTTL3
TTL4	The switch asserts TTL4 each time a path is created.	
	C	IVISWTCH_VAL_TTL4
	COM	IviSwrchAdvancedOutputTTL4
TTL5	The switch asserts TTL5 each time a path is created.	
	C	IVISWTCH_VAL_TTL5
	COM	IviSwrchAdvancedOutputTTL5
TTL6	The switch asserts TTL6 each time a path is created.	
	C	IVISWTCH_VAL_TTL6
	COM	IviSwrchAdvancedOutputTTL6
TTL7	The switch asserts TTL7 each time a path is created.	
	C	IVISWTCH_VAL_TTL7
	COM	IviSwrchAdvancedOutputTTL7
ECL0	The switch asserts ECL0 each time a path is created.	
	C	IVISWTCH_VAL_ECL0
	COM	IviSwrchAdvancedOutputECL0
ECL1	The switch asserts ECL1 each time a path is created.	
	C	IVISWTCH_VAL_ECL1
	COM	IviSwrchAdvancedOutputECL1
PXI Star	The switch asserts PXI Star each time a path is created.	
	C	IVISWTCH_VAL_PXI_STAR
	COM	IviSwrchAdvancedOutputPXIStar
RTSI 0	The switch asserts RTSI0 each time a path is created.	
	C	IVISWTCH_VAL_RTSI_0
	COM	IviSwrchAdvancedOutputRTSI0
RTSI 1	The switch asserts RTSI1 each time a path is created.	
	C	IVISWTCH_VAL_RTSI_1
	COM	IviSwrchAdvancedOutputRTSI1

Name	Description	
	Language	Identifier
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`.

5.2.6 Scan Delay

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	N/A	None	Configure Scan Trigger

COM Property Name

Scan.Delay

COM Enumeration Name

N/A

C Constant Name

IVISWTCH_ATTR_SCAN_DELAY

Description

Specifies the *minimum* length of time, in milliseconds, 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: The unit for Scan Delay is milliseconds, not seconds.

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

5.2.7 Scan List

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

COM Property Name

Scan.List

COM Enumeration Name

N/A

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:

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.
;	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.

Symbol	Symbol Name	Syntax Example	Description
~	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 .

5.2.8 Scan Mode

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

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 trigger event (i.e., before the semi-colon) should be broken, and if so, how they should be broken.

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.	
	C	IVISWTCH_VAL_BREAK_BEFORE_MAKE
	COM	IviSwTchScanModeBreakBeforeMake
Break After Make	Tells the driver to make new paths before breaking the previous paths.	
	C	IVISWTCH_VAL_BREAK_AFTER_MAKE
	COM	IviSwTchScanModeBreakAfterMake

<i>Name</i>	<i>Description</i>	
	<i>Language</i>	<i>Identifier</i>
None	Indicates that no action should be taken on the previous paths.	
	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.

5.2.9 Trigger Input

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

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

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
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

Name	Description	
	Language	Identifier
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.	
	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

Name	Description	
	Language	Identifier
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
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	IviSwchSoftwareTrigger

See Section 8, IviSwch 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`.

5.3 IviSwchScanner Functions

The IviSwchScanner 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.

COM Prototype

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

C Prototype

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

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession

Return Values

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.

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.

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.

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

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.

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.

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
TriggerInput	Trigger input. The driver uses this value to set the Trigger Input attribute. See the attribute description for more details.	ViInt32
AdvancedOutput	Scan advanced output. The driver uses this value to set the Scan Advanced Output attribute. See the attribute description for more details.	ViInt32

Return Values

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.

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.

COM Prototype

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

C Prototype

```
ViStatus IviSwch_InitiateScan (ViSession Vi);
```

Parameters

Inputs	Description	Data Type
vi	Instrument handle	ViSession

Return Values

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.

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.

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

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.

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

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.

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

Return Values

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.

5.4 IviSwthScanner 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 IviSwth 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.

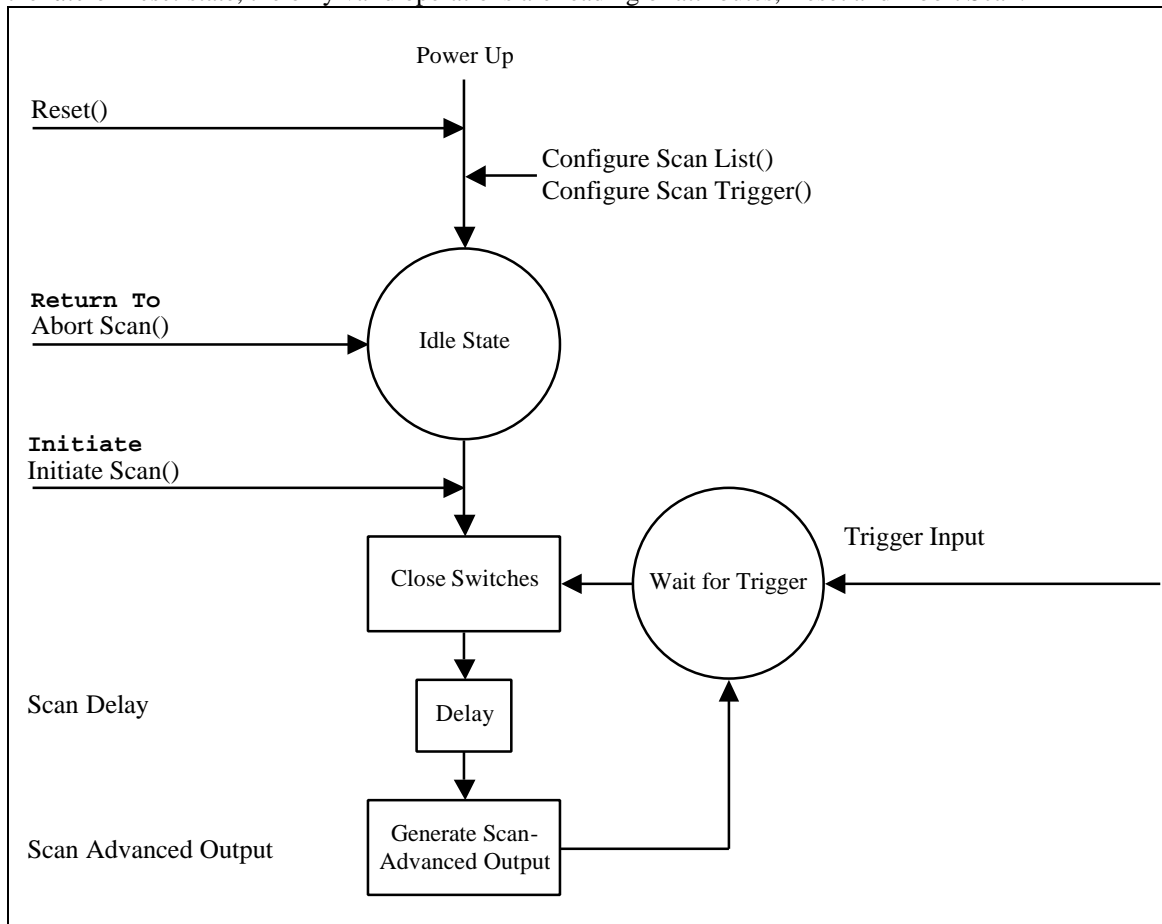


Figure 5-1. IviSwth Trigger Model

6. IviSwchSoftwareTrigger Extension Group

6.1 IviSwchSoftwareTrigger Overview

The IviSwchSoftwareTrigger 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 IviSwchSoftwareTrigger Functions

The IviSwchSoftwareTrigger 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.

COM Prototype

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

C Prototype

```
ViStatus IviSwTch_SendSoftwareTrigger (ViSession vi);
```

Parameters

Inputs	Description	Data Type
vi	Instrument handle	ViSession

Return Values

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.

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

<i>Value Name</i>	<i>Language</i>	<i>Identifier</i>	<i>Actual Value</i>
None	C	IVISWTCH_VAL_NONE	0
	COM	IviSwTchScanModeNone	0
Break Before Make	C	IVISWTCH_VAL_BREAK_BEFORE_MAKE	1
	COM	IviSwTchScanModeBreakBeforeMake	1
Break After Make	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

Trigger Input

<i>Value Name</i>	<i>Language</i>	<i>Identifier</i>	<i>Actual Value</i>
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
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

<i>Value Name</i>	<i>Language</i>	<i>Identifier</i>	<i>Actual Value</i>
TTL5	C	IVISWTCH_VAL_TTL5	116
	COM	IviSwchTriggerInputTTL5	116
TTL6	C	IVISWTCH_VAL_TTL6	117
	COM	IviSwchTriggerInputTTL6	117
TTL7	C	IVISWTCH_VAL_TTL7	118
	COM	IviSwchTriggerInputTTL7	118
ECL0	C	IVISWTCH_VAL_ECL0	119
	COM	IviSwchTriggerInputECL0	119
ECL1	C	IVISWTCH_VAL_ECL1	120
	COM	IviSwchTriggerInputECL1	120
PXI Star	C	IVISWTCH_VAL_PXI_STAR	125
	COM	IviSwchTriggerInputPXIStar	125
RTSI 0	C	IVISWTCH_VAL_RTSI_0	140
	COM	IviSwchTriggerInputRTSI0	140
RTSI 1	C	IVISWTCH_VAL_RTSI_1	141
	COM	IviSwchTriggerInputRTSI1	141
RTSI 2	C	IVISWTCH_VAL_RTSI_2	142
	COM	IviSwchTriggerInputRTSI2	142
RTSI 3	C	IVISWTCH_VAL_RTSI_3	143
	COM	IviSwchTriggerInputRTSI3	143
RTSI 4	C	IVISWTCH_VAL_RTSI_4	144
	COM	IviSwchTriggerInputRTSI4	144
RTSI 5	C	IVISWTCH_VAL_RTSI_5	145
	COM	IviSwchTriggerInputRTSI5	145
RTSI 6	C	IVISWTCH_VAL_RTSI_6	146
	COM	IviSwchTriggerInputRTSI6	146
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

<i>Value Name</i>	<i>Language</i>	<i>Identifier</i>	<i>Actual Value</i>
None	C	IVISWTCH_VAL_NONE	0
	COM	IviSwchAdvancedOutputNone	0

<i>Value Name</i>	<i>Language</i>	<i>Identifier</i>	<i>Actual Value</i>
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
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

<i>Value Name</i>	<i>Language</i>	<i>Identifier</i>	<i>Actual Value</i>
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

<i>Value Name</i>	<i>Language</i>	<i>Identifier</i>	<i>Actual Value</i>
Path Available	C	IVISWTCH_VAL_PATH_AVAILABLE	1
	COM	IviSwTchPathAvailable	1
Path Exists	C	IVISWTCH_VAL_PATH_EXISTS	2
	COM	IviSwTchPathExists	2
Path Unsupported	C	IVISWTCH_VAL_PATH_UNSUPPORTED	3
	COM	IviSwTchPathUnsupported	3
Resource In Use	C	IVISWTCH_VAL_RSRC_IN_USE	4
	COM	IviSwTchPathRsrcInUse	4
Source Conflict	C	IVISWTCH_VAL_SOURCE_CONFLICT	5
	COM	IviSwTchPathSourceConflict	5
Channel Not Available	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. IviSwch Error and Completion Code Value Definitions

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

Table 10-1. IviSwch Error and Completion Codes

<i>Error Name</i>	<i>Description</i>		
	<i>Language</i>	<i>Identifier</i>	<i>Value(hex)</i>
Path Remains	Some connections remain after disconnecting.		
	C	IVISWTCH_WARN_PATH_REMAINS	0x3FFFA2001
	COM	S_IVISWTCH_PATH_REMAINS	0x00042001
Implicit Connection Exists	The implicit connection exists between the channels.		
	C	IVISWTCH_WARN_IMPLICIT_CONNECTION_EXISTS	0x3FFFA2002
	COM	S_IVISWTCH_IMPLICIT_CONNECTION_EXISTS	0x00042002
Trigger Not Software	The trigger source is not set to software trigger.		
	C	IVISWTCH_ERROR_TRIGGER_NOT_SOFTWARE	0xBFFFA1001
	COM	E_IVISWTCH_TRIGGER_NOT_SOFTWARE	0x80041001
Invalid Switch Path	Invalid path list string.		
	C	IVISWTCH_ERROR_INVALID_SWITCH_PATH	0xBFFFA2001
	COM	E_IVISWTCH_INVALID_SWITCH_PATH	0x80042001
Invalid Scan List	The given scan list string does not have the correct syntax.		
	C	IVISWTCH_ERROR_INVALID_SCAN_LIST	0xBFFFA2002
	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.		
	C	IVISWTCH_ERROR_RSRC_IN_USE	0xBFFFA2003
	COM	E_IVISWTCH_RSRC_IN_USE	0x80042003
Empty Scan List	No scan list specified.		
	C	IVISWTCH_ERROR_EMPTY_SCAN_LIST	0xBFFFA2004
	COM	E_IVISWTCH_EMPTY_SCAN_LIST	0x80042004
Empty Switch Path	The specified path list string is empty.		
	C	IVISWTCH_ERROR_EMPTY_SWITCH_PATH	0xBFFFA2005
	COM	E_IVISWTCH_EMPTY_SWITCH_PATH	0x80042005

Table 10-1. IviSwTch Error and Completion Codes

<i>Error Name</i>	<i>Description</i>		
	<i>Language</i>	<i>Identifier</i>	<i>Value(hex)</i>
Scan In Progress	The switch module is currently scanning through the scan list.		
	C	IVISWTCH_ERROR_SCAN_IN_PROGRESS	0xBFFFA2006
	COM	E_IVISWTCH_SCAN_IN_PROGRESS	0x80042006
No Scan In Progress	The switch module is not currently scanning through the scan list.		
	C	IVISWTCH_ERROR_NO_SCAN_IN_PROGRESS	0xBFFFA2007
	COM	E_IVISWTCH_NO_SCAN_IN_PROGRESS	0x80042007
No Such Path	No explicit path exists between the channels.		
	C	IVISWTCH_ERROR_NO_SUCH_PATH	0xBFFFA2008
	COM	E_IVISWTCH_NO_SUCH_PATH	0x80042008
Is Configuration Channel	An explicit connection to a configuration channel is not allowed.		
	C	IVISWTCH_ERROR_IS_CONFIGURATION_CHANNEL	0xBFFFA2009
	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.		
	C	IVISWTCH_ERROR_NOT_A_CONFIGURATION_CHANNEL	0xBFFFA200A
	COM	E_IVISWTCH_NOT_A_CONFIGURATION_CHANNEL	0x8004200A
Attempt To Connect Sources	A connection between two different sources is not allowed.		
	C	IVISWTCH_ERROR_ATTEMPT_TO_CONNECT_SOURCES	0xBFFFA200B
	COM	E_IVISWTCH_ATTEMPT_TO_CONNECT_SOURCES	0x8004200B
Explicit Connection Exists	An explicit connection between the channels already exists.		
	C	IVISWTCH_ERROR_EXPLICIT_CONNECTION_EXISTS	0xBFFFA200C
	COM	E_IVISWTCH_EXPLICIT_CONNECTION_EXISTS	0x8004200C
Leg Missing First Channel	A leg in the path does not begin with a channel name.		
	C	IVISWTCH_ERROR_LEG_MISSING_FIRST_CHANNEL	0xBFFFA200D
	COM	E_IVISWTCH_LEG_MISSING_FIRST_CHANNEL	0x8004200D

Table 10-1. IviSwTch Error and Completion Codes

<i>Error Name</i>	<i>Description</i>		
	<i>Language</i>	<i>Identifier</i>	<i>Value(hex)</i>
Leg Missing Second Channel	A leg in the path is missing the second channel.		
	C	IVISWTCH_ERROR_LEG_MISSING_SECOND_CHANNEL	0xBFFA200E
Channel Duplicated In Leg	COM	E_IVISWTCH_LEG_MISSING_SECOND_CHANNEL	0x8004200E
	The first and the second channels in the leg are the same.		
Channel Duplicated In Path	C	IVISWTCH_ERROR_CHANNEL_DUPLICATED_IN_LEG	0xBFFA200F
	COM	E_IVISWTCH_CHANNEL_DUPLICATED_IN_LEG	0x8004200F
Path Not Found	A channel name is duplicated in the path string.		
	C	IVISWTCH_ERROR_CHANNEL_DUPLICATED_IN_PATH	0xBFFA2010
Discontinuous Path	COM	E_IVISWTCH_CHANNEL_DUPLICATED_IN_PATH	0x80042010
	No path was found between the two channels.		
Cannot Connect Directly	C	IVISWTCH_ERROR_PATH_NOT_FOUND	0xBFFA2011
	COM	E_IVISWTCH_PATH_NOT_FOUND	0x80042011
Channels Already Connected	The first channel of a leg in the path is not the same as the second channel in the previous leg.		
	C	IVISWTCH_ERROR_DISCONTINUOUS_PATH	0xBFFA2012
Channels Already Connected	COM	E_IVISWTCH_DISCONTINUOUS_PATH	0x80042012
	The path contains a leg with two channels that cannot be directly connected.		
Channels Already Connected	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.		
	C	IVISWTCH_ERROR_CHANNELS_ALREADY_CONNECTED	0xBFFA2014
	COM	E_IVISWTCH_CHANNELS_ALREADY_CONNECTED	0x80042014

Table 10-1. IviSwTch Error and Completion Codes

<i>Error Name</i>	<i>Description</i>		
	<i>Language</i>	<i>Identifier</i>	<i>Value(hex)</i>
Cannot Connect To Itself	A channel cannot be connected to itself.		
	C	IVISWTCH_ERROR_CANNOT_CONNECT_TO_ITSELF	0xBFFFA2015
	COM	E_IVISWTCH_CANNOT_CONNECT_TO_ITSELF	0x80042015
Max Time Exceeded	Maximum time exceeded before the operation completed.		
	C	IVISWTCH_ERROR_MAX_TIME_EXCEEDED	0xBFFFA2016
	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.

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”
Invalid Scan List	“%s: Invalid scan list”
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”

Table 10-2. IviSwtch Error Message Strings

Name	Message String
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"

11. IviSwch Hierarchies

11.1 IviSwch COM Hierarchy

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

Table 11-1. IviSwch 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
GetPath	Get Path	M
SetPath	Set Path	M

Table 11-1. IviSwch COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
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

11.1.1 IviSwch COM Interfaces

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

1. IiviSwchPath
2. IiviSwchScan
3. IiviSwchChannels

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

The IiviSwchChannel interface contains an interface reference property for accessing the IiviSwchCharacteristics interface.

Table 11-2. IviSwch Interface GUIDs lists the interfaces that this specification defines and their GUIDs.

Table 11-2. IviSwch Interface GUIDs

Interface	GUID
IiviSwch	47ed527e-a398-11d4-ba58-000064657374
IiviSwchPath	47ed527f-a398-11d4-ba58-000064657374
IiviSwchScan	47ed5280-a398-11d4-ba58-000064657374
IiviSwchChannels	47ed5281-a398-11d4-ba58-000064657374
IiviSwchChannel	47ed5282-a398-11d4-ba58-000064657374
IiviSwchCharacteristics	47ed5283-a398-11d4-ba58-000064657374

11.1.2 Interface Reference Properties

Interface reference properties are used to navigate the IviSwch COM hierarchy. This section describes the interface reference properties that the IiviSwch and IiviSwchChannel interfaces define.

11.1.2.1 Channels

Data Type	Access
IiviSwchChannels*	RO

COM Property Name

Channels

Description

Returns a pointer to the IiviSwchChannels interface.

11.1.2.2 Scan

Data Type	Access
IiviSwchScan*	RO

COM Property Name

Scan

Description

Returns a pointer to the IiviSwchScan interface.

11.1.2.3 Path

Data Type	Access
IiviSwchPath*	RO

COM Property Name

Path

Description

Returns a pointer to the IiviSwchPath interface.

11.1.2.4 Characteristics

Data Type	Access
IIviSwchCharacteristics*	RO

COM Property Name

Channel.Characteristics

Description

Returns a pointer to the IIviSwchCharacteristics interface.

11.1.3 IviSwtch COM Category

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

11.2 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 4.2, *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
<i>Configuration...</i>	
Configure Scan List	IviSwtch_ConfigureScanList
Configure Scan Trigger	IviSwtch_ConfigureScanTrigger
Set Continuous Scan	IviSwtch_SetContinuousScan
<i>Route...</i>	
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
<i>Paths...</i>	
Set Path	IviSwtch_SetPath
Get Path	IviSwtch_GetPath
<i>Scan...</i>	
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
<i>Utility...</i>	
Get Channel Name	IviSwtch_GetChannelName

11.3 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 4.2, *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
<i>Channel Configuration</i>	
Is Source Channel	IVISWTCH_ATTR_IS_SOURCE_CHANNEL
Is Configuration Channel	IVISWTCH_ATTR_IS_CONFIGURATION_CHANNEL

Table 11-5. IviSwTch C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
<i>Module Characteristics</i>	
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
<i>Scanning Configuration</i>	
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
<i>Matrix Configuration</i>	
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 IviSwch 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 IviSwchSoftwareTrigger Extension Group

The IviSwchSoftwareTrigger 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 IviSwchSoftwareTrigger 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.

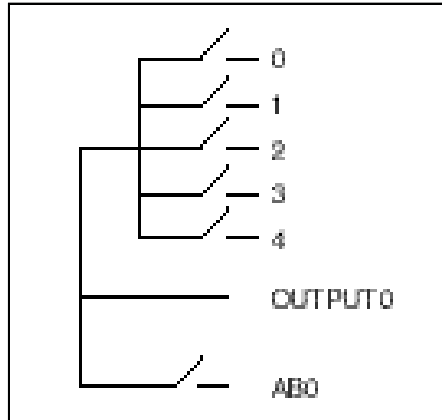


Figure A-1. Analog Bus Example

As you can see from Figure 3, to connect 1 to ABO 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 IviSwch 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 ABO, 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. IviSwch 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 than 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 IviSwch 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 IviSwch instrument driver that operates on smaller IviSwch 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 IviSwch instrument driver definition does not provide a way to access these configuration switches in an interoperable fashion. In these cases, an IviSwch 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 IviSwch class has been designed primarily to handle routing and scanning issues that face more complex switch systems. However, the IviSwch 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`.

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 where 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 IviSwchScanner 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.

IviSwchBase Capability Group

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

IviSwchScanner Capability Group

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

Appendix C. ANSI C Include File

```
/*
 *
 * Title:      IviSwTch include file
 * Purpose:    IviSwTch Class declarations for Inherent Capabilities and
 *            IviSwTch Base and Extended Capabilities.
 */
*****

#ifndef IVISWTCH_HEADER
#define IVISWTCH_HEADER

#if defined(__cplusplus) || defined(__cplusplus)
extern "C" {
#endif

/*
 *----- IviSwTch Class Attribute Defines -----*
 */
/*- IviSwTchBase Attributes -*/
#define IVISWTCH_ATTR_CHANNEL_COUNT          IVI_INHERENT_ATTR_BASE + 203

#define IVISWTCH_ATTR_IS_SOURCE_CHANNEL      (IVI_CLASS_ATTR_BASE + 1)
#define IVISWTCH_ATTR_IS_DEBOUNCED         (IVI_CLASS_ATTR_BASE + 2)
#define IVISWTCH_ATTR_IS_CONFIGURATION_CHANNEL (IVI_CLASS_ATTR_BASE + 3)
#define IVISWTCH_ATTR_MAX_SETTLING_TIME    (IVI_CLASS_ATTR_BASE + 4)
#define IVISWTCH_ATTR_BANDWIDTH            (IVI_CLASS_ATTR_BASE + 5)
#define IVISWTCH_ATTR_MAX_DC_VOLTAGE      (IVI_CLASS_ATTR_BASE + 6)
#define IVISWTCH_ATTR_MAX_AC_VOLTAGE      (IVI_CLASS_ATTR_BASE + 7)
#define IVISWTCH_ATTR_MAX_SWITCHING_DC_CURRENT (IVI_CLASS_ATTR_BASE + 8)
#define IVISWTCH_ATTR_MAX_SWITCHING_AC_CURRENT (IVI_CLASS_ATTR_BASE + 9)
#define IVISWTCH_ATTR_MAX_CARRY_DC_CURRENT (IVI_CLASS_ATTR_BASE + 10)
#define IVISWTCH_ATTR_MAX_CARRY_AC_CURRENT (IVI_CLASS_ATTR_BASE + 11)
#define IVISWTCH_ATTR_MAX_SWITCHING_DC_POWER (IVI_CLASS_ATTR_BASE + 12)
#define IVISWTCH_ATTR_MAX_SWITCHING_AC_POWER (IVI_CLASS_ATTR_BASE + 13)
#define IVISWTCH_ATTR_MAX_CARRY_DC_POWER (IVI_CLASS_ATTR_BASE + 14)
#define IVISWTCH_ATTR_MAX_CARRY_AC_POWER (IVI_CLASS_ATTR_BASE + 15)
#define IVISWTCH_ATTR_CHARACTERISTIC_IMPEDANCE (IVI_CLASS_ATTR_BASE + 16)
#define IVISWTCH_ATTR_WIRE_MODE           (IVI_CLASS_ATTR_BASE + 17)

/*- IviSwTch Extended Attributes -*/
/*- IviSwitchScanner Extension Group -*/
#define IVISWTCH_ATTR_NUM_OF_ROWS          (IVI_CLASS_ATTR_BASE + 18)
#define IVISWTCH_ATTR_NUM_OF_COLUMNS      (IVI_CLASS_ATTR_BASE + 19)
#define IVISWTCH_ATTR_SCAN_LIST           (IVI_CLASS_ATTR_BASE + 20)
#define IVISWTCH_ATTR_SCAN_MODE           (IVI_CLASS_ATTR_BASE + 21)
#define IVISWTCH_ATTR_TRIGGER_INPUT       (IVI_CLASS_ATTR_BASE + 22)
#define IVISWTCH_ATTR_SCAN_ADVANCED_OUTPUT (IVI_CLASS_ATTR_BASE + 23)
#define IVISWTCH_ATTR_IS_SCANNING         (IVI_CLASS_ATTR_BASE + 24)
#define IVISWTCH_ATTR_SCAN_DELAY          (IVI_CLASS_ATTR_BASE + 25)
#define IVISWTCH_ATTR_CONTINUOUS_SCAN     (IVI_CLASS_ATTR_BASE + 26)

/*
 *----- IviSwTch Class Attribute Value Defines -----*
 */
/*- Defined values for attribute IVISWTCH_ATTR_SCAN_MODE -*/
#define IVISWTCH_VAL_NONE                  (0)
#define IVISWTCH_VAL_BREAK_BEFORE_MAKE    (1)
#define IVISWTCH_VAL_BREAK_AFTER_MAKE     (2)
#define IVISWTCH_VAL_SCAN_MODE_CLASS_EXT_BASE (500)
#define IVISWTCH_VAL_SCAN_MODE_SPECIFIC_EXT_BASE (1000)

/*- Defined values for attribute IVISWTCH_ATTR_TRIGGER_INPUT -*/
#define IVISWTCH_VAL_IMMEDIATE            (1)
#define IVISWTCH_VAL_EXTERNAL             (2)

```

```

#define IVISWTCH_VAL_SOFTWARE_TRIG (3)
#define IVISWTCH_VAL_TTL0 (111)
#define IVISWTCH_VAL_TTL1 (112)
#define IVISWTCH_VAL_TTL2 (113)
#define IVISWTCH_VAL_TTL3 (114)
#define IVISWTCH_VAL_TTL4 (115)
#define IVISWTCH_VAL_TTL5 (116)
#define IVISWTCH_VAL_TTL6 (117)
#define IVISWTCH_VAL_TTL7 (118)
#define IVISWTCH_VAL_ECL0 (119)
#define IVISWTCH_VAL_ECL1 (120)
#define IVISWTCH_VAL_PXI_STAR (125)
#define IVISWTCH_VAL_RTSI_0 (140)
#define IVISWTCH_VAL_RTSI_1 (141)
#define IVISWTCH_VAL_RTSI_2 (142)
#define IVISWTCH_VAL_RTSI_3 (143)
#define IVISWTCH_VAL_RTSI_4 (144)
#define IVISWTCH_VAL_RTSI_5 (145)
#define IVISWTCH_VAL_RTSI_6 (146)
#define IVISWTCH_VAL_TRIGGER_INPUT_CLASS_EXT_BASE (500)
#define IVISWTCH_VAL_TRIGGER_INPUT_SPECIFIC_EXT_BASE (1000)

/*- Defined values for attribute IVISWTCH_ATTR_SCAN_ADVANCED_OUTPUT -*/
#define IVISWTCH_VAL_GPIB_SRQ (5)
/* #define IVISWTCH_VAL_NONE DEFINED ABOVE */
/* #define IVISWTCH_VAL_EXTERNAL DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL0 DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL1 DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL2 DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL3 DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL4 DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL5 DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL6 DEFINED ABOVE */
/* #define IVISWTCH_VAL_TTL7 DEFINED ABOVE */
/* #define IVISWTCH_VAL_ECL0 DEFINED ABOVE */
/* #define IVISWTCH_VAL_ECL1 DEFINED ABOVE */
/* #define IVISWTCH_VAL_PXI_STAR DEFINED ABOVE */
/* #define IVISWTCH_VAL_RTSI_0 DEFINED ABOVE */
/* #define IVISWTCH_VAL_RTSI_1 DEFINED ABOVE */
/* #define IVISWTCH_VAL_RTSI_2 DEFINED ABOVE */
/* #define IVISWTCH_VAL_RTSI_3 DEFINED ABOVE */
/* #define IVISWTCH_VAL_RTSI_4 DEFINED ABOVE */
/* #define IVISWTCH_VAL_RTSI_5 DEFINED ABOVE */
/* #define IVISWTCH_VAL_RTSI_6 DEFINED ABOVE */
#define IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_CLASS_EXT_BASE (500)
#define IVISWTCH_VAL_SCAN_ADVANCED_OUTPUT_SPECIFIC_EXT_BASE (1000)

/*- Defined values for IviSwTch_CanConnect path capability parameter -*/
#define IVISWTCH_VAL_PATH_AVAILABLE (1)
#define IVISWTCH_VAL_PATH_EXISTS (2)
#define IVISWTCH_VAL_PATH_UNSUPPORTED (3)
#define IVISWTCH_VAL_RSRC_IN_USE (4)
#define IVISWTCH_VAL_SOURCE_CONFLICT (5)
#define IVISWTCH_VAL_CHANNEL_NOT_AVAILABLE (6)
#define IVISWTCH_VAL_CAN_CONNECT_CLASS_EXT_BASE (500)
#define IVISWTCH_VAL_CAN_CONNECT_SPECIFIC_EXT_BASE (1000)

/*****
 *----- IviSwTch Class Instrument Driver Function Declarations -----*
 *****/

/* Utility */

ViStatus _VI_FUNC IviSwTch_GetChannelName (ViSession Vi,
                                           ViInt32 Index,
                                           ViInt32 NameBufferSize,
                                           ViChar Name[]);

/*- IviSwTchBase Capability Group Functions -*/

ViStatus _VI_FUNC IviSwTch_CanConnect (ViSession Vi,

```

```

        ViConstString Channel1,
        ViConstString Channel2,
        ViInt32 *PathCapability);

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

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

ViStatus _VI_FUNC IviSwtch_DisconnectAll (ViSession Vi);

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

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

ViStatus _VI_FUNC IviSwtch_SetPath (ViSession Vi,
        ViConstString PathList);

ViStatus _VI_FUNC IviSwtch_WaitForDebounce (ViSession Vi,
        ViInt32 MaxTimeMilliseconds);

/*- IviSwtchScanner Extension Group Functions -*/
ViStatus _VI_FUNC IviSwtch_AbortScan (ViSession Vi);

ViStatus _VI_FUNC IviSwtch_ConfigureScanList (ViSession Vi,
        ViConstString List,
        ViInt32 Mode);

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

ViStatus _VI_FUNC IviSwtch_InitiateScan (ViSession Vi);

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

ViStatus _VI_FUNC IviSwtch_SetContinuousScan (ViSession Vi,
        ViBoolean Status);

ViStatus _VI_FUNC IviSwtch_WaitForScanComplete (ViSession Vi,
        ViInt32 MaxTimeMilliseconds);

/*- IviSwtchSoftwareTrigger Extension Group Functions -*/
ViStatus _VI_FUNC IviSwtch_SendSoftwareTrigger (ViSession Vi);

/*****
 *----- IviSwtch Class Error And Completion Codes -----*
 *****/
#define IVISWTCH_WARN_PATH_REMAINS (IVI_CLASS_WARN_BASE + 1)
#define IVISWTCH_WARN_IMPLICIT_CONNECTION_EXISTS (IVI_CLASS_WARN_BASE + 2)

#define IVISWTCH_ERROR_INVALID_SWITCH_PATH (IVI_CLASS_ERROR_BASE + 1)
#define IVISWTCH_ERROR_INVALID_SCAN_LIST (IVI_CLASS_ERROR_BASE + 2)
#define IVISWTCH_ERROR_RSRC_IN_USE (IVI_CLASS_ERROR_BASE + 3)
#define IVISWTCH_ERROR_EMPTY_SCAN_LIST (IVI_CLASS_ERROR_BASE + 4)
#define IVISWTCH_ERROR_EMPTY_SWITCH_PATH (IVI_CLASS_ERROR_BASE + 5)
#define IVISWTCH_ERROR_SCAN_IN_PROGRESS (IVI_CLASS_ERROR_BASE + 6)
#define IVISWTCH_ERROR_NO_SCAN_IN_PROGRESS (IVI_CLASS_ERROR_BASE + 7)
#define IVISWTCH_ERROR_NO_SUCH_PATH (IVI_CLASS_ERROR_BASE + 8)
#define IVISWTCH_ERROR_IS_CONFIGURATION_CHANNEL (IVI_CLASS_ERROR_BASE + 9)
#define IVISWTCH_ERROR_NOT_A_CONFIGURATION_CHANNEL (IVI_CLASS_ERROR_BASE + 10)

```

```

#define IVISWTCH_ERROR_ATTEMPT_TO_CONNECT_SOURCES (IVI_CLASS_ERROR_BASE + 11)
#define IVISWTCH_ERROR_EXPLICIT_CONNECTION_EXISTS (IVI_CLASS_ERROR_BASE + 12)
#define IVISWTCH_ERROR_LEG_MISSING_FIRST_CHANNEL (IVI_CLASS_ERROR_BASE + 13)
#define IVISWTCH_ERROR_LEG_MISSING_SECOND_CHANNEL (IVI_CLASS_ERROR_BASE + 14)
#define IVISWTCH_ERROR_CHANNEL_DUPLICATED_IN_LEG (IVI_CLASS_ERROR_BASE + 15)
#define IVISWTCH_ERROR_CHANNEL_DUPLICATED_IN_PATH (IVI_CLASS_ERROR_BASE + 16)
#define IVISWTCH_ERROR_PATH_NOT_FOUND (IVI_CLASS_ERROR_BASE + 17)
#define IVISWTCH_ERROR_DISCONTINUOUS_PATH (IVI_CLASS_ERROR_BASE + 18)
#define IVISWTCH_ERROR_CANNOT_CONNECT_DIRECTLY (IVI_CLASS_ERROR_BASE + 19)
#define IVISWTCH_ERROR_CHANNELS_ALREADY_CONNECTED (IVI_CLASS_ERROR_BASE + 20)
#define IVISWTCH_ERROR_CANNOT_CONNECT_TO_ITSELF (IVI_CLASS_ERROR_BASE + 21)
#define IVISWTCH_ERROR_MAX_TIME_EXCEEDED (IVI_CLASS_ERROR_BASE + 22)

#define IVISWTCH_ERROR_TRIGGER_NOT_SOFTWARE (IVI_SHARED_COMPONENT_ERROR_BASE + 1)

/*****
 *----- End Include File -----*
 *****/
#if defined(__cplusplus) || defined(_cplusplus)
}
#endif
#endif /* IVISWTCH_HEADER */

```

Appendix D. COM IDL File

To ease the development of a compliant an IVI-COM driver for the IviSwtch class, the IVI Foundation publishes IDL (Interface Description Language) files that consolidate all the method and property definitions listed in this specification. Notice that the interface IviSwtch derives from IiviDriver, described in IVI-3.2, *Inherent Capabilities Specification*.

These files along with these definitions compiled into type libraries are available from the IVI Foundation web site at <http://www.ivifoundation.org/>.

D.1 *IviSwtchTypeLib.idl*

This file contains the type library definitions.

```
#if !defined(IVI_SWTCH_TYPELIB_IDL_INCLUDED_)
#define IVI_SWTCH_TYPELIB_IDL_INCLUDED_
/*****
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 * All rights reserved.
 *
 *
 * FILENAME      : IviSwtchTypeLib.idl
 *
 * STATUS        : APPROVED.
 * COMPILER      : MSVC++ 6.0, sp4 MIDL
 * CONTENT       : IVI Swtch Instrument Class Standard IDL
 *               : type library definition
 *****/

import "oidl.idl";
import "ocidl.idl";

[
    uuid(47ed5125-a398-11d4-ba58-000064657374),
    version(3.0),
    helpstring("IviSwtch 3.0 Type Library"),
    helpfile("IviSwtch.chm")
]
library IviSwtchLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

#include "IviSwtch.idl"
};

#endif // !defined(IVI_SWTCH_TYPELIB_IDL_INCLUDED_)
```

D.2 IviSwtch.idl

This file contains interface definitions, functions prototypes, and enumerations.

```
#if !defined(IVI_SWTCH_IDL_INCLUDED_)
#define IVI_SWTCH_IDL_INCLUDED_
/*****
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 * All rights reserved.
 *
 * FILENAME      : IviSwtch.idl
 *
 * STATUS        : APPROVED.
 * COMPILER      : MSVC++ 6.0, sp4 MIDL
 * CONTENT       : IVI Swtch Instrument Class Standard IDL
 *
 *****/

#include <winerror.h>
import "oidl.idl";
import "ocidl.idl";

#ifdef IVI_USE_IMPORT
import "..\IviDriverTypeLib\IviDriver.idl";
#else
importlib ("..\TypeLibraries\IviDriverTypeLib.dll");
#endif

//-----
// Preprocessor Macros
//-----

#define HELP_SWTCH(x)  helpstring(HS_SWTCH_ ## x ## ), helpcontext(HC_SWTCH_ ## x ## )

//-----
// Provides for Localization
//-----

#include "IviSwtchEnglish.idl"

//-----
// Interface Declarations
//-----

interface IIviSwtch;
interface IIviSwtchPath;
interface IIviSwtchScan;
interface IIviSwtchChannels;
interface IIviSwtchChannel;
interface IIviSwtchCharacteristics;
```

```

#define UUID_IIVI_SWTCH 47ed527e-a398-11d4-ba58-000064657374
#define UUID_IIVI_SWTCH_PATH 47ed527f-a398-11d4-ba58-000064657374
#define UUID_IIVI_SWTCH_SCAN 47ed5280-a398-11d4-ba58-000064657374
#define UUID_IIVI_SWTCH_CHANNELS 47ed5281-a398-11d4-ba58-000064657374
#define UUID_IIVI_SWTCH_CHANNEL 47ed5282-a398-11d4-ba58-000064657374
#define UUID_IIVI_SWTCH_CHARACTERISTICS 47ed5283-a398-11d4-ba58-000064657374

//-----
// TYPEDEF ENUMS
//-----

[
    HELP_SWTCH(HRESULTS_ENUM)
]
typedef enum IviSwTchErrorCodesEnum
{
    E_IVISWTCH_TRIGGER_NOT_SOFTWARE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x1001),
    E_IVISWTCH_INVALID_SWITCH_PATH = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2001),
    E_IVISWTCH_INVALID_SCAN_LIST = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2002),
    E_IVISWTCH_RSRC_IN_USE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2003),
    E_IVISWTCH_EMPTY_SCAN_LIST = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2004),
    E_IVISWTCH_EMPTY_SWITCH_PATH = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2005),
    E_IVISWTCH_SCAN_IN_PROGRESS = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2006),
    E_IVISWTCH_NO_SCAN_IN_PROGRESS = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2007),
    E_IVISWTCH_NO_SUCH_PATH = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2008),
    E_IVISWTCH_IS_CONFIGURATION_CHANNEL = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2009),
    E_IVISWTCH_NOT_A_CONFIGURATION_CHANNEL = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x200A),
    E_IVISWTCH_ATTEMPT_TO_CONNECT_SOURCES = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x200B),
    E_IVISWTCH_EXPLICIT_CONNECTION_EXISTS = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x200C),
    E_IVISWTCH_LEG_MISSING_FIRST_CHANNEL = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x200D),
    E_IVISWTCH_LEG_MISSING_SECOND_CHANNEL = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x200E),
    E_IVISWTCH_CHANNEL_DUPLICATED_IN_LEG = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x200F),
    E_IVISWTCH_CHANNEL_DUPLICATED_IN_PATH = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2010),

```

```

    E_IVISWTCH_PATH_NOT_FOUND           = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2011),
    E_IVISWTCH_DISCONTINUOUS_PATH       = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2012),
    E_IVISWTCH_CANNOT_CONNECT_DIRECTLY  = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2013),
    E_IVISWTCH_CHANNELS_ALREADY_CONNECTED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2014),
    E_IVISWTCH_CANNOT_CONNECT_TO_ITSELF = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2015),
    E_IVISWTCH_MAX_TIME_EXCEEDED        = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x2016),
    S_IVISWTCH_PATH_REMAINS             = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF,
0x2001),
    S_IVISWTCH_IMPLICIT_CONNECTION_EXISTS = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF,
0x2002)
} IviSwTchErrorCodesEnum;

```

```

//-----
//  enum: IviSwTchAdvancedOutput
//-----

```

```

typedef
[
    public,
    v1_enum,
    HELP_SWTCH(ADVANCED_OUTPUT_ENUM)
]
enum IviSwTchAdvancedOutputEnum
{
    IviSwTchAdvancedOutputNone           = 0,
    IviSwTchAdvancedOutputExternal       = 2,
    IviSwTchAdvancedOutputGPIBSRQ       = 5,
    IviSwTchAdvancedOutputTTL0           = 111,
    IviSwTchAdvancedOutputTTL1           = 112,
    IviSwTchAdvancedOutputTTL2           = 113,
    IviSwTchAdvancedOutputTTL3           = 114,
    IviSwTchAdvancedOutputTTL4           = 115,
    IviSwTchAdvancedOutputTTL5           = 116,
    IviSwTchAdvancedOutputTTL6           = 117,
    IviSwTchAdvancedOutputTTL7           = 118,
    IviSwTchAdvancedOutputECL0           = 119,
    IviSwTchAdvancedOutputECL1           = 120,
    IviSwTchAdvancedOutputPXISStar       = 125,
    IviSwTchAdvancedOutputRTSI0          = 140,
    IviSwTchAdvancedOutputRTSI1          = 141,
    IviSwTchAdvancedOutputRTSI2          = 142,
    IviSwTchAdvancedOutputRTSI3          = 143,
    IviSwTchAdvancedOutputRTSI4          = 144,
    IviSwTchAdvancedOutputRTSI5          = 145,
    IviSwTchAdvancedOutputRTSI6          = 146
} IviSwTchAdvancedOutputEnum;

```

```

//-----
//  enum: IviSwrchPathCapability
//-----

typedef
[
    public,
    vl_enum,
    HELP_SWTCH(PATH_CAPABILITY_ENUM)
]
enum IviSwrchPathCapabilityEnum
{
    IviSwrchPathAvailable           = 1,
    IviSwrchPathExists              = 2,
    IviSwrchPathUnsupported         = 3,
    IviSwrchPathRsrcInUse           = 4,
    IviSwrchPathSourceConflict      = 5,
    IviSwrchPathChannelNotAvailable = 6
} IviSwrchPathCapabilityEnum;

//-----
//  enum: IviSwrchScanMode
//-----

typedef
[
    public,
    vl_enum,
    HELP_SWTCH(SCAN_MODE_ENUM)
]
enum IviSwrchScanModeEnum
{
    IviSwrchScanModeNone           = 0,
    IviSwrchScanModeBreakBeforeMake = 1,
    IviSwrchScanModeBreakAfterMake = 2
} IviSwrchScanModeEnum;

//-----
//  enum: IviSwrchTriggerInput
//-----

typedef
[
    public,
    vl_enum,
    HELP_SWTCH(TRIGGER_INPUT_ENUM)
]
enum IviSwrchTriggerInputEnum
{
    IviSwrchTriggerInputImmediate   = 1,
    IviSwrchTriggerInputExternal    = 2,
}

```

```

    IviSwchTriggerInputSwTrigFunc          = 3,
    IviSwchTriggerInputTTL0                = 111,
    IviSwchTriggerInputTTL1                = 112,
    IviSwchTriggerInputTTL2                = 113,
    IviSwchTriggerInputTTL3                = 114,
    IviSwchTriggerInputTTL4                = 115,
    IviSwchTriggerInputTTL5                = 116,
    IviSwchTriggerInputTTL6                = 117,
    IviSwchTriggerInputTTL7                = 118,
    IviSwchTriggerInputECL0                = 119,
    IviSwchTriggerInputECL1                = 120,
    IviSwchTriggerInputPXIStar             = 125,
    IviSwchTriggerInputRTSI0               = 140,
    IviSwchTriggerInputRTSI1               = 141,
    IviSwchTriggerInputRTSI2               = 142,
    IviSwchTriggerInputRTSI3               = 143,
    IviSwchTriggerInputRTSI4               = 144,
    IviSwchTriggerInputRTSI5               = 145,
    IviSwchTriggerInputRTSI6               = 146
} IviSwchTriggerInputEnum;

//-----
//  IVI Swtch Driver Root Level Interface
//-----

[
    object,
    uuid(UUID_IIVI_SWTCH),
    HELP_SWTCH(I_IIVI_SWTCH),
    oleautomation,
    pointer_default(unique)
]
interface IIviSwtch : IIviDriver
{
//----- Channels Interface Reference
    [ propget, HELP_SWTCH(CHANNELS) ]
    HRESULT Channels([out, retval] IIviSwtchChannels **pVal);

//----- Path Interface Reference
    [ propget, HELP_SWTCH(PATH) ]
    HRESULT Path([out, retval] IIviSwtchPath **pVal);

//----- Scan Interface Reference
    [ propget, HELP_SWTCH(SCAN) ]
    HRESULT Scan([out, retval] IIviSwtchScan **pVal);

};

//-----

```

```

// Channels Interface
//-----

[
    object,
    uuid(UUID_IIVI_SWTCH_CHANNELS),
    HELP_SWTCH(I_IVI_SWTCH_CHANNELS),
    oleautomation,
    pointer_default(unique)
]
interface IIVI_SwtchChannels : IUnknown
{
//----- Item
    [ propget, HELP_SWTCH(CHANNELS_ITEM) ]
    HRESULT Item (
        [in] BSTR Name,
        [out, retval] IIVI_SwtchChannel **pVal);

//----- Count
    [ propget, HELP_SWTCH(CHANNELS_COUNT) ]
    HRESULT Count ([out, retval] LONG *pVal);

//----- Name
    [ propget, HELP_SWTCH(CHANNELS_NAME) ]
    HRESULT Name ([in] LONG Index, [out, retval] BSTR *Name);
};

//-----
// Channel Interface
//-----

[
    object,
    uuid(UUID_IIVI_SWTCH_CHANNEL),
    HELP_SWTCH(I_IVI_SWTCH_CHANNEL),
    oleautomation,
    pointer_default(unique)
]
interface IIVI_SwtchChannel : IUnknown
{
//----- IsConfigurationChannel
    [ propput, HELP_SWTCH(IS_CONFIGURATION_CHANNEL) ]
    HRESULT IsConfigurationChannel([in] VARIANT_BOOL newVal);

    [ propget, HELP_SWTCH(IS_CONFIGURATION_CHANNEL) ]
    HRESULT IsConfigurationChannel([out, retval] VARIANT_BOOL *pVal);

//----- IsSourceChannel
    [ propput, HELP_SWTCH(IS_SOURCE_CHANNEL) ]

```

```

HRESULT IsSourceChannel([in] VARIANT_BOOL newVal);

[ propget, HELP_SWTCH(IS_SOURCE_CHANNEL) ]
HRESULT IsSourceChannel([out, retval] VARIANT_BOOL *pVal);

//----- Characteristics Interface Reference
[ propget, HELP_SWTCH(CHARACTERISTICS) ]
HRESULT Characteristics([out, retval] IIVI_SwtchCharacteristics **pVal);

};

//-----
// Characteristics Interface
//-----

[
    object,
    uuid(UUID_IIVI_SWTCH_CHARACTERISTICS),
    HELP_SWTCH(I_IVI_SWTCH_CHARACTERISTICS),
    oleautomation,
    pointer_default(unique)
]
interface IIVI_SwtchCharacteristics : IUnknown
{
//----- ACCurrentCarryMax
[ propget, HELP_SWTCH(AC_CURRENT_CARRY_MAX) ]
HRESULT ACCurrentCarryMax([out, retval] DOUBLE *pVal);

//----- ACCurrentSwitchingMax
[ propget, HELP_SWTCH(AC_CURRENT_SWITCHING_MAX) ]
HRESULT ACCurrentSwitchingMax([out, retval] DOUBLE *pVal);

//----- ACPowerCarryMax
[ propget, HELP_SWTCH(AC_POWER_CARRY_MAX) ]
HRESULT ACPowerCarryMax([out, retval] DOUBLE *pVal);

//----- ACPowerSwitchingMax
[ propget, HELP_SWTCH(AC_POWER_SWITCHING_MAX) ]
HRESULT ACPowerSwitchingMax([out, retval] DOUBLE *pVal);

//----- ACVoltageMax
[ propget, HELP_SWTCH(AC_VOLTAGE_MAX) ]
HRESULT ACVoltageMax([out, retval] DOUBLE *pVal);

//----- Bandwidth
[ propget, HELP_SWTCH(BANDWIDTH) ]
HRESULT Bandwidth([out, retval] DOUBLE *pVal);

```

```

//----- DCCurrentCarryMax
[ propget, HELP_SWTCH(DC_CURRENT_CARRY_MAX) ]
HRESULT DCCurrentCarryMax([out, retval] DOUBLE *pVal);

//----- DCCurrentSwitchingMax
[ propget, HELP_SWTCH(DC_CURRENT_SWITCHING_MAX) ]
HRESULT DCCurrentSwitchingMax([out, retval] DOUBLE *pVal);

//----- DCPowerCarryMax
[ propget, HELP_SWTCH(DC_POWER_CARRY_MAX) ]
HRESULT DCPowerCarryMax([out, retval] DOUBLE *pVal);

//----- DCPowerSwitchingMax
[ propget, HELP_SWTCH(DC_POWER_SWITCHING_MAX) ]
HRESULT DCPowerSwitchingMax([out, retval] DOUBLE *pVal);

//----- DCVoltageMax
[ propget, HELP_SWTCH(DC_VOLTAGE_MAX) ]
HRESULT DCVoltageMax([out, retval] DOUBLE *pVal);

//----- Impedance
[ propget, HELP_SWTCH(IMPEDANCE) ]
HRESULT Impedance([out, retval] DOUBLE *pVal);

//----- SettlingTime
[ propget, HELP_SWTCH(SETTLING_TIME) ]
HRESULT SettlingTime([out, retval] DOUBLE *pVal);

//----- WireMode
[ propget, HELP_SWTCH(WIRE_MODE) ]
HRESULT WireMode([out, retval] LONG *pVal);

};

//-----
// Path Interface
//-----

[
    object,
    uuid(UUID_IIVI_SWTCH_PATH),
    HELP_SWTCH(I_IVI_SWTCH_PATH),
    oleautomation,
    pointer_default(unique)
]

```

```

]
interface IIVI_Switch_Path : IUnknown
{
//----- CanConnect
[ HELP_SWTCH(CAN_CONNECT) ]
HRESULT CanConnect(
    [in] BSTR Channel1,
    [in] BSTR Channel2,
    [out,retval] IIVI_Switch_Path_CapabilityEnum *PathCapability);

//----- Connect
[ HELP_SWTCH(CONNECT) ]
HRESULT Connect(
    [in] BSTR Channel1,
    [in] BSTR Channel2);

//----- Disconnect
[ HELP_SWTCH(DISCONNECT) ]
HRESULT Disconnect(
    [in] BSTR Channel1,
    [in] BSTR Channel2);

//----- DisconnectAll
[ HELP_SWTCH(DISCONNECT_ALL) ]
HRESULT DisconnectAll();

//----- GetPath
[ HELP_SWTCH(GET_PATH) ]
HRESULT GetPath(
    [in] BSTR Channel1,
    [in] BSTR Channel2,
    [out, retval] BSTR *PathList);

//----- IsDebounced
[ propget, HELP_SWTCH(IS_DEBOUNCED) ]
HRESULT IsDebounced([out, retval] VARIANT_BOOL *pVal);

//----- SetPath
[ HELP_SWTCH(SET_PATH) ]
HRESULT SetPath(
    [in] BSTR PathList);

//----- WaitForDebounce
[ HELP_SWTCH(WAIT_FOR_DEBOUNCE) ]
HRESULT WaitForDebounce(
    [in] LONG MaxTimeMilliseconds);

```

```

};

//-----
//  Scan Interface
//-----

[
    object,
    uuid(UUID_IIVI_SWTCH_SCAN),
    HELP_SWTCH(I_IVI_SWTCH_SCAN),
    oleautomation,
    pointer_default(unique)
]
interface IIVI_SwtchScan : IUnknown
{
//----- ConfigureList
    [ HELP_SWTCH(CONFIGURE_LIST) ]
    HRESULT ConfigureList(
        [in] BSTR List,
        [in] IIVI_SwtchScanModeEnum Mode);

//----- ConfigureTrigger
    [ HELP_SWTCH(CONFIGURE_TRIGGER) ]
    HRESULT ConfigureTrigger(
        [in] DOUBLE ScanDelay,
        [in] IIVI_SwtchTriggerInputEnum TriggerInput,
        [in] IIVI_SwtchAdvancedOutputEnum AdvancedOutput);

//----- Initiate
    [ HELP_SWTCH(INITIATE) ]
    HRESULT Initiate();

//----- Abort
    [ HELP_SWTCH(ABORT) ]
    HRESULT Abort();

//----- IsScanning
    [ propget, HELP_SWTCH(IS_SCANNING) ]
    HRESULT IsScanning([out, retval] VARIANT_BOOL *pVal);

//----- SendSoftwareTrigger
    [ HELP_SWTCH(SEND_SOFTWARE_TRIGGER) ]
    HRESULT SendSoftwareTrigger();

//----- WaitForScanComplete
    [ HELP_SWTCH(WAIT_FOR_SCAN_COMPLETE) ]
    HRESULT WaitForScanComplete([in] LONG MaxTimeMilliseconds);
}

```

```

//----- Continuous
[ propget, HELP_SWTCH(CONTINUOUS) ]
HRESULT Continuous([in] VARIANT_BOOL newVal);

[ propget, HELP_SWTCH(CONTINUOUS) ]
HRESULT Continuous([out, retval] VARIANT_BOOL *pVal);

//----- NumberOfColumns
[ propget, HELP_SWTCH(NUMBER_OF_COLUMNS) ]
HRESULT NumberOfColumns([out, retval] LONG *pVal);

//----- NumberOfRows
[ propget, HELP_SWTCH(NUMBER_OF_ROWS) ]
HRESULT NumberOfRows([out, retval] LONG *pVal);

//----- AdvancedOutput
[ propget, HELP_SWTCH(ADVANCED_OUTPUT) ]
HRESULT AdvancedOutput([in] IviSwTchAdvancedOutputEnum newVal);

[ propget, HELP_SWTCH(ADVANCED_OUTPUT) ]
HRESULT AdvancedOutput([out, retval] IviSwTchAdvancedOutputEnum *pVal);

//----- Delay
[ propget, HELP_SWTCH(DELAY) ]
HRESULT Delay([in] DOUBLE newVal);

[ propget, HELP_SWTCH(DELAY) ]
HRESULT Delay([out, retval] DOUBLE *pVal);

//----- Input
[ propget, HELP_SWTCH(TRIGGER_INPUT) ]
HRESULT Input([in] IviSwTchTriggerInputEnum newVal);

[ propget, HELP_SWTCH(TRIGGER_INPUT) ]
HRESULT Input([out, retval] IviSwTchTriggerInputEnum *pVal);

//----- List
[ propget, HELP_SWTCH(LIST) ]
HRESULT List([in] BSTR newVal);

[ propget, HELP_SWTCH(LIST) ]
HRESULT List([out, retval] BSTR *pVal);

//----- Mode
[ propget, HELP_SWTCH(MODE) ]

```

```

    HRESULT Mode([in] IviSwtchScanModeEnum newVal);

    [ propget, HELP_SWTCH(MODE) ]
    HRESULT Mode([out, retval] IviSwtchScanModeEnum *pVal);

};

#endif // !defined(IVI_SWTCH_IDL_INCLUDED_)

```

D.3 *IviSwtchEnglish.idl*

This file contains help strings and contexts for the interfaces, methods and properties.

```

#if !defined(IVI_SWTCH_IDL_ENGLISH_INCLUDED_)
#define IVI_SWTCH_IDL_ENGLISH_INCLUDED_
/*****
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 * All rights reserved.
 *
 *
 * FILENAME      : IviSwtchEnglish.idl
 *
 * STATUS        : APPROVED.
 * COMPILER      : MSVC++ 6.0, sp4 MIDL
 * CONTENT       : IVI Swtch Instrument Class Standard IDL
 *                help context IDs and help strings
 *
 *****/

#define HC_SWTCH_BASE 600

//-----
//      TYPEDEF ENUMS
//-----

#define HC_SWTCH_HRESULTS_ENUM                HC_SWTCH_BASE + 0

#define HC_SWTCH_ADVANCED_OUTPUT_ENUM        HC_SWTCH_BASE + 1
#define HC_SWTCH_PATH_CAPABILITY_ENUM        HC_SWTCH_BASE + 2
#define HC_SWTCH_SCAN_MODE_ENUM              HC_SWTCH_BASE + 3
#define HC_SWTCH_TRIGGER_INPUT_ENUM          HC_SWTCH_BASE + 4

//-----
//      IVI Switch Driver Root Level Interface
//-----

#define HC_SWTCH_I_IVI_SWTCH                  HC_SWTCH_BASE + 10
#define HC_SWTCH_CHANNELS                      HC_SWTCH_BASE + 11
#define HC_SWTCH_PATH                          HC_SWTCH_BASE + 12
#define HC_SWTCH_SCAN                          HC_SWTCH_BASE + 13

```

```

//-----
//  Channels Interface
//-----

#define HC_SWTCH_I_IVI_SWTCH_CHANNELS          HC_SWTCH_BASE + 14
#define HC_SWTCH_CHANNELS_ITEM                HC_SWTCH_BASE + 15
#define HC_SWTCH_CHANNELS_COUNT              HC_SWTCH_BASE + 16
#define HC_SWTCH_CHANNELS_NAME               HC_SWTCH_BASE + 17

//-----
//  Channel Interface
//-----

#define HC_SWTCH_I_IVI_SWTCH_CHANNEL          HC_SWTCH_BASE + 18
#define HC_SWTCH_IS_CONFIGURATION_CHANNEL    HC_SWTCH_BASE + 19
#define HC_SWTCH_IS_SOURCE_CHANNEL          HC_SWTCH_BASE + 20
#define HC_SWTCH_CHARACTERISTICS            HC_SWTCH_BASE + 21

//-----
//  Characteristics Interface
//-----

#define HC_SWTCH_I_IVI_SWTCH_CHARACTERISTICS  HC_SWTCH_BASE + 22
#define HC_SWTCH_AC_CURRENT_CARRY_MAX       HC_SWTCH_BASE + 23
#define HC_SWTCH_AC_CURRENT_SWITCHING_MAX   HC_SWTCH_BASE + 24
#define HC_SWTCH_AC_POWER_CARRY_MAX        HC_SWTCH_BASE + 25
#define HC_SWTCH_AC_POWER_SWITCHING_MAX    HC_SWTCH_BASE + 26
#define HC_SWTCH_AC_VOLTAGE_MAX            HC_SWTCH_BASE + 27
#define HC_SWTCH_BANDWIDTH                  HC_SWTCH_BASE + 28
#define HC_SWTCH_DC_CURRENT_CARRY_MAX      HC_SWTCH_BASE + 29
#define HC_SWTCH_DC_POWER_CARRY_MAX        HC_SWTCH_BASE + 30
#define HC_SWTCH_DC_CURRENT_SWITCHING_MAX  HC_SWTCH_BASE + 31
#define HC_SWTCH_DC_POWER_SWITCHING_MAX    HC_SWTCH_BASE + 32
#define HC_SWTCH_DC_VOLTAGE_MAX            HC_SWTCH_BASE + 33
#define HC_SWTCH_IMPEDANCE                  HC_SWTCH_BASE + 34
#define HC_SWTCH_SETTLING_TIME              HC_SWTCH_BASE + 35
#define HC_SWTCH_WIRE_MODE                  HC_SWTCH_BASE + 36

//-----
//  Path Interface
//-----

#define HC_SWTCH_I_IVI_SWTCH_PATH           HC_SWTCH_BASE + 37
#define HC_SWTCH_CAN_CONNECT                HC_SWTCH_BASE + 38
#define HC_SWTCH_CONNECT                    HC_SWTCH_BASE + 39
#define HC_SWTCH_DISCONNECT                 HC_SWTCH_BASE + 40
#define HC_SWTCH_DISCONNECT_ALL             HC_SWTCH_BASE + 41
#define HC_SWTCH_GET_PATH                   HC_SWTCH_BASE + 42
#define HC_SWTCH_IS_DEBOUNCED              HC_SWTCH_BASE + 43

```

```

#define HC_SWTCH_SET_PATH                HC_SWTCH_BASE + 44
#define HC_SWTCH_WAIT_FOR_DEBOUNCE      HC_SWTCH_BASE + 45

```

```

//-----
//   Scan Interface
//-----

```

```

#define HC_SWTCH_I_IVI_SWTCH_SCAN        HC_SWTCH_BASE + 46
#define HC_SWTCH_CONFIGURE_LIST          HC_SWTCH_BASE + 47
#define HC_SWTCH_CONFIGURE_TRIGGER       HC_SWTCH_BASE + 48
#define HC_SWTCH_INITIATE                 HC_SWTCH_BASE + 49
#define HC_SWTCH_ABORT                    HC_SWTCH_BASE + 50
#define HC_SWTCH_IS_SCANNING              HC_SWTCH_BASE + 51
#define HC_SWTCH_SEND_SOFTWARE_TRIGGER    HC_SWTCH_BASE + 52
#define HC_SWTCH_WAIT_FOR_SCAN_COMPLETE   HC_SWTCH_BASE + 53
#define HC_SWTCH_CONTINUOUS               HC_SWTCH_BASE + 54
#define HC_SWTCH_NUMBER_OF_COLUMNS        HC_SWTCH_BASE + 55
#define HC_SWTCH_NUMBER_OF_ROWS           HC_SWTCH_BASE + 56
#define HC_SWTCH_ADVANCED_OUTPUT          HC_SWTCH_BASE + 57
#define HC_SWTCH_DELAY                     HC_SWTCH_BASE + 58
#define HC_SWTCH_TRIGGER_INPUT            HC_SWTCH_BASE + 59
#define HC_SWTCH_LIST                      HC_SWTCH_BASE + 60
#define HC_SWTCH_MODE                      HC_SWTCH_BASE + 61

```

```

//-----
//   TYPEDEF ENUMS
//-----

```

```

#define HS_SWTCH_HRESULTS_ENUM \
    "IVI Swtch class defined HRESULTS"

#define HS_SWTCH_ADVANCED_OUTPUT_ENUM \
    "IVI Swtch class-compliant values for scan Advanced Output"

#define HS_SWTCH_PATH_CAPABILITY_ENUM \
    "IVI Swtch class-compliant values for PathCapability"

#define HS_SWTCH_SCAN_MODE_ENUM \
    "IVI Swtch class-compliant values for Scan Mode"

#define HS_SWTCH_TRIGGER_INPUT_ENUM \
    "IVI Swtch class-compliant values for Trigger Input"

```

```

//-----
//   IVI Switch Driver Root Level Interface
//-----

```

```

#define HS_SWTCH_I_IVI_SWTCH \
    "IVI Swtch class-compliant root level interface"

#define HS_SWTCH_CHANNELS \

```

```

"Pointer to the class-compliant IIVI_SwtchChannels interface"

#define HS_SWTCH_PATH \
"Pointer to the class-compliant IIVI_SwtchPath interface"

#define HS_SWTCH_SCAN \
"Pointer to the class-compliant IIVI_SwtchScan interface"

//-----
// Channels Interface
//-----

#define HS_SWTCH_I_IVI_SWTCH_CHANNELS \
"IVI Swtch class-compliant channel collection interface"

#define HS_SWTCH_CHANNELS_ITEM \
"An interface reference pointer to one of the IVI Swtch Channel interfaces \
which is selected by the channel name."

#define HS_SWTCH_CHANNELS_COUNT \
"The number of channels."

#define HS_SWTCH_CHANNELS_NAME \
"The channel name for a given index."

//-----
// Channel Interface
//-----

#define HS_SWTCH_I_IVI_SWTCH_CHANNEL \
"IVI Swtch class-compliant channel interface"

#define HS_SWTCH_IS_CONFIGURATION_CHANNEL \
"If True, the specific driver can use the channel for internal path \
creation. The only operations that may be performed on configuration \
channels are reading and writing the IsConfigurationChannel property."

#define HS_SWTCH_IS_SOURCE_CHANNEL \
"If True, the channel is a source channel. Two channels that are either \
sources or have their own connections to sources, may not be connected. \
This prevents connections that might cause damage to channels, devices, \
or system."

#define HS_SWTCH_CHARACTERISTICS \
"Pointer to the class-compliant IIVI_SwtchCharacteristics interface"

//-----
// Characteristics Interface
//-----

#define HS_SWTCH_I_IVI_SWTCH_CHARACTERISTICS \

```

```

"IVI Swtch class-compliant channel characteristics interface"

#define HS_SWTCH_AC_CURRENT_CARRY_MAX \
"The maximum AC current the channel can carry, in amperes."

#define HS_SWTCH_AC_CURRENT_SWITCHING_MAX \
"The maximum AC current the channel can switch, in amperes."

#define HS_SWTCH_AC_POWER_CARRY_MAX \
"The maximum AC power the channel can handle, in volt-amperes."

#define HS_SWTCH_AC_POWER_SWITCHING_MAX \
"The maximum AC power the channel can switch, in volt-amperes."

#define HS_SWTCH_AC_VOLTAGE_MAX \
"The maximum AC voltage the channel can handle, in volts RMS."

#define HS_SWTCH_BANDWIDTH \
"The bandwidth, in Hertz, for the channel."

#define HS_SWTCH_DC_CURRENT_CARRY_MAX \
"The maximum DC current the channel can carry, in amperes."

#define HS_SWTCH_DC_CURRENT_SWITCHING_MAX \
"The maximum DC current the channel can switch, in amperes."

#define HS_SWTCH_DC_POWER_CARRY_MAX \
"The maximum DC power the channel can handle, in watts."

#define HS_SWTCH_DC_POWER_SWITCHING_MAX \
"The maximum DC power the channel can switch, in watts."

#define HS_SWTCH_DC_VOLTAGE_MAX \
"The maximum DC voltage the channel can handle, in volts."

#define HS_SWTCH_IMPEDANCE \
"The characteristic impedance of the channel, in ohms."

#define HS_SWTCH_SETTLING_TIME \
"The maximum total settling time, in seconds, 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."

#define HS_SWTCH_WIRE_MODE \
"The number of conductors in the current channel."

//-----
// Path Interface
//-----

#define HS_SWTCH_I_IVI_SWTCH_PATH \
"IVI Swtch class-compliant path interface"

```

```

#define HS_SWTCH_CAN_CONNECT \
"Without actually creating the path, reports whether the switch module can \
create a given path. This method takes existing paths into account."

#define HS_SWTCH_CONNECT \
"Creates a path between the two channels. If the path already exists, \
the operation does not count the number of calls. All paths are assumed to \
be bi-directional."

#define HS_SWTCH_DISCONNECT \
"Destroys the path between the two channels. The order of the two channels \
need not be the same as the connect operation."

#define HS_SWTCH_DISCONNECT_ALL \
"Disconnect all paths created since Initialize or Reset have been called. \
If any paths remain, this method returns a Path Remains warning."

#define HS_SWTCH_GET_PATH \
"Returns the list of comma separated channel pairs that have been connected \
in order to create the path between the specified channels. The only valid \
paths that can be returned are ones that have been explicitly set via \
Connect and SetPath methods."

#define HS_SWTCH_IS_DEBOUNCED \
"If True, the switch module has settled from previous switching commands \
and completed debounce. This indicates that the signal going through the \
switch module is valid, assuming that the switches in the path have the \
correct characteristics."

#define HS_SWTCH_SET_PATH \
"Creates a path given a PathList of comma separated channel pairs. The end \
channels are the first and last entries in the PathList. For intermediate \
channels, the second channel of one pair must be the same as the first \
channel of the following pair."

#define HS_SWTCH_WAIT_FOR_DEBOUNCE \
"Wait until all the signals flowing through the switch have settled. \
If the signals do not settle within MaxTime, the method returns a \
Max Time Exceeded error."

//-----
//  Scan Interface
//-----

#define HS_SWTCH_I_IVI_SWTCH_SCAN \
"IVI Swtch class-compliant scan interface"

#define HS_SWTCH_CONFIGURE_LIST \
"Configures the scan List and Mode properties."

#define HS_SWTCH_CONFIGURE_TRIGGER \
"Configures the scan Delay, Input trigger, and AdvancedOutput trigger \

```

```

properties, in order to set the trigger behavior for the scan list \
established with the ConfigureList method."

#define HS_SWTCH_INITIATE \
"Initiates the scan based on the current List property, and returns once \
the scan has begun. To stop scanning, call Abort. When the switch \
is scanning, operations other than reading properties, SendSoftwareTrigger \
and Abort are invalid."

#define HS_SWTCH_ABORT \
"Stops the scan begun with Initiate method and returns the switch to the \
Idle state. This operation does not reset or initialize the state of the \
switch. To determine the status of the scan, call the IsScanning \
method."

#define HS_SWTCH_IS_SCANNING \
"If True, the switch module is currently scanning through the scan list."

#define HS_SWTCH_SEND_SOFTWARE_TRIGGER \
"Sends a software trigger, which causes the switch to advance to the next \
entry in the scan List."

#define HS_SWTCH_WAIT_FOR_SCAN_COMPLETE \
"Waits, up to MaxTime milliseconds, for the instrument to stop scanning \
through the scan List."

#define HS_SWTCH_CONTINUOUS \
"If True, the switch module should scan continuously through the scan \
list, otherwise it should only scan once through the scan list."

#define HS_SWTCH_NUMBER_OF_COLUMNS \
"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 input \
channels. The number returned is dependent on the WireMode property."

#define HS_SWTCH_NUMBER_OF_ROWS \
"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 output \
channels (commons). The number returned is dependent on the WireMode \
property."

#define HS_SWTCH_ADVANCED_OUTPUT \
"The destination of the scan's advanced output trigger. This trigger \
is asserted each time a path is created."

#define HS_SWTCH_DELAY \
"The minimum length of time, in milliseconds, from creating the path to \
asserting the scan's advanced output trigger. The actual time may be longer."

#define HS_SWTCH_TRIGGER_INPUT \
"The source of the trigger input. Input triggers tell the switch module to \
advance to the next entry in the scan list and close the specified channel."

#define HS_SWTCH_LIST \

```

```
"A list of the channels to scan, and the order to scan them in. The format \
of this string is described in the IviSwTch specification."
```

```
#define HS_SWTCH_MODE \  
"The connection breaking behavior of the switch during a scan. Previous \  
paths may be broken before making new ones, after making new ones, or not \  
broken at all."
```

```
#endif // !defined(IVI_SWTCH_IDL_ENGLISH_INCLUDED_)
```