IVI-4.1: IviScope Class Specification

April 2009 Edition
Revision 3.0

Important Information

The IviScope Class Specification (IVI-4.1) is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at `www.ivifoundation.org`.

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

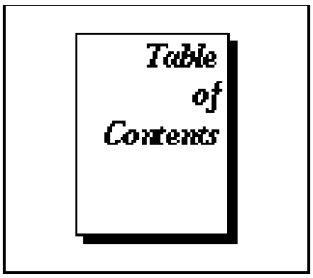
Warranty

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

Trademarks

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

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



IviScope Class Specification

IviScope Revision History

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

Table 1. IviScope Class Specification Revisions

Revision Number	Date of Revision	Revision Notes
Revision 0.1	July 30, 1997	Original draft.
Revision 0.2	August 12, 1997	This edition incorporates edits based on user feedback and adds introductory text.
Revision 0.3	October 28, 1997	This edition incorporates edits to clean up inconsistencies in terminology and addition descriptive text.
Revision 0.4	December 2, 1997	Reformatted document. Created advanced triggering extension groups
Revision 0.5	February 6, 1998	Reformatted document. Created extension group that reads min/max waveforms that the oscilloscope acquires when it is in average or peak detect mode.
Revision 1.0	August 21, 1998	Technical Publications review and edit. Changes to template information.
Revision 2.0b	July 20, 1999	Changes to the specification as the result of the Vendors' comments.
Revision 2.0c	September 01, 1999	Reformatted the document. Removed the Miscellaneous extension group and distributed the attributes in their own extension groups or in the base capability group. Renamed the Fundamental capability group to base capabilities group. Added the Interpolation extension group Added the AC Line triggering extension group Added the Probe auto-sense extension group Added the Continuous Acquisition extension group Added the Average extension group Added the Sample Mode extension group Added the Trigger Modifier extension group Added the Auto-setup extension group. Changed the compliance information for attributes and functions.
Revision 2.0d	October 7, 1999	Changed the actual value for the `IVISCOPE_VAL_NEGATIVE` from 2 to 0. Adjusted other values of attribute Ids and attribute value sets to maintain backwards compatibility for the drivers that

Table 1. IviScope Class Specification Revisions

		were written to the 1.0 version of this specification. Removed Software Triggering.
Revision 2.0	November 22, 1999	Changed the Spec number to 2.0. Removed the "DRAFT" characterization—the specification has been approved.
Revision 3.0a	December 2000	Reformatted to adhere to IVI-3.4: Elements of Style for Class Specifications spec. Also specified the COM interface. Changed the specification number to 4.1.
Revision 3.0b	March 13, 2001	Implemented changes as agreed at the Feb 2001 IVI-COM working group meeting.
Revision 2.1 (vc1)	July 1st, 2001	Added appendices with C header and COM IDL file contents. Fixed miscellaneous typos and style issues.
Revision 2.1 (vc2)	August 31, 2001	Acted upon feedback from reviewers
Revision 2.1 (vc3)	December 18, 2001	Implemented changes as agreed at the December IVI Foundation Inc. meeting.
Revision 2.1 (vc3.1)	December 27, 2001	Changed section references, corrected actual AttributeID for the channel count attribute, and removed the capability reference for the channel count properties.
Revision 3.0 (vc3.2)	January 21, 2002	Syntax check, formatting etc. Important change: removed the C description for ATTR_MEASUREMENT_CHANNEL_COUNT attribute since it is a COM –only property.
Revision 3.0 (vc3.3)	February 28, 2002	Final fixes to the IDL appendix, MaxTime parameter name change to MaxTimeMilliseconds, and changing the method prototypes to use LONG instead of IviScopeMaxTimeEnum for the MaxTimeMilliseconds parameter only. Added GUIDs for the IIviScopeChannel and IIviScopeMeasurement interfaces.
Revision 3.0	April 04, 2002	Specification has been approved – updated TOC, typelib version number, and updated COM UUIDs to release version.
Revision 3.0	February 14, 2003	Corrected a typo on page 95 (changed wording of ATTR_POLARITY to ATTR_RUNT_POLARITY)
Revision 3.0	October 01, 2004	Editorial change: Added the function prototype for the IviScope_GetChannelName function to the Appendix C: ANSI C Include file; this function exists in the specification (section 4.3.11) but was omitted from the appendix by mistake.
Revision 3.0	April 29, 2008	Editorial change to update the IVI Foundation contact information in the Important Information section to remove obsolete address information and refer only to the IVI Foundation web site.
Revision 3.0	April 2009	Editorial change to update repeated capabilities section to include both qualified and unqualified

Table 1. IviScope Class Specification Revisions

		repeated capability names.

1. Overview of the IviScope Specification

1.1 Introduction

This specification defines the IVI class for oscilloscopes. The IviScope class is designed to support the typical oscilloscope as well as common extended functionality found in more complex instruments. This section summarizes the *IviScope 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:

- IviScope Class Overview
- References
- The definitions of Terms and Acronyms

1.2 IviScope Class Overview

This specification defines the IVI class for oscilloscopes called IviScope. The IviScope class is designed to support the typical oscilloscope as well as common extended functionality found in more complex instruments. The IviScope class conceptualizes an oscilloscope as an instrument that can acquire time varying voltage waveforms.

The IviScope class is divided into the base capability group and extensions. The base capability group functions and attributes are used to configure an oscilloscope for typical waveform acquisition (this includes setting the channel, the acquisition, and the triggering sub-systems), initiating the waveform acquisition, and returning a waveform. The base capability group support only edge triggering and normal waveform acquisition. The IviScopeBase Capabilities are described in Section 4: *IviScopeBase Capability Group*.

In addition to the base capabilities, the IviScope class defines extended capabilities for oscilloscopes that can:

- Interpolate the points in the waveform record
- Have advanced triggering options such as TV, runt, glitch, width, and AC line
- Sense the probe attenuation
- Perform an auto-setup
- Use alternative acquisition modes such as average, envelope, and peak detect
- Use different sample modes such as real-time and equivalent
- Acquire data continuously
- Perform waveform measurements such as rise-time, fall-time, and voltage peak-to-peak

The IviScope extended capabilities are arranged into a set of extension capability groups.

1.3 References

The following documents and specifications are related to this specification:

- IVI-3.1: Driver Architecture Specification
- IVI-3.12: Floating Point Services Specification
- IVI-3.2: Inherent Capabilities Specification
- IVI-3.4: API Style Guide
- IVI-5.0: Glossary

1.4 Definitions of Terms and Acronyms

Refer to *IVI-5.0: Glossary* for a description of the terms and acronyms used in this specification. This specification does not define any additional terms.

2. IviScope Class Capabilities

2.1 Introduction

The IviScope specification divides generic oscilloscope 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 IviScope Group Names

The capability group names for the IviScope 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. IviScope Group Names

Group Name	Description
IviScopeBase	Base Capabilities of the IviScope specification. This group includes the capability to acquire waveforms using edge triggering.
IviScopeInterpolation	Extension: IviScope with the ability to configure the oscilloscope to interpolate missing points in a waveform.
IviScopeTVTrigger	Extension: IviScope with the ability to trigger on standard television signals.
IviScopeRuntTrigger	Extension: IviScope with the ability to trigger on runts.
IviScopeGlitchTrigger	Extension: IviScope with the ability to trigger on glitches.
IviScopeWidthTrigger	Extension: IviScope with the ability to trigger on a variety of conditions regarding pulse widths.
IviScopeAcLineTrigger	Extension: IviScope with the ability to trigger on zero crossings of a network supply voltage.
IviScopeWaveformMeas	Extension: IviScope with the ability to calculate waveform measurements, such as rise time or frequency.
IviScopeMinMaxWaveform	Extension: IviScope with the ability to acquire a minimum and maximum waveforms that correspond to the same time range.
IviScopeProbeAutoSense	Extension: IviScope with the ability to automatically sense the probe attenuation of an attached probe.
IviScopeContinuous Acquisition	Extension: IviScope with the ability to continuously acquire data from the input and display it on the screen.
IviScopeAverage Acquisition	Extension: IviScope with the ability to create a waveform that is the average of multiple waveform acquisitions.
IviScopeSampleMode	Extension: IviScope with the ability to return the actual sample mode.
IviScopeTrigger Modifier	Extension: IviScope with the ability to modify the behavior of the triggering subsystem in the absence of a expected trigger.
IviScopeAutoSetup	Extension: IviScope with the automatic configuration ability.

Refer to Section 15, *Class Specification Layout*, in *IVI-3.4: API Style Guide* for a description of the Capability Group Section Layout.

2.3 Repeated Capability Names

The IviScope Class Specification defines one repeated capability. 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*.

- Channel

2.3.1 Channel

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

3. General Requirements

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

3.1 Minimum Class Compliance

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

3.1.1 Disable

Refer to *IVI-3.2: Inherent Capabilities Specification* for the prototype of this function. The IviScope specification does not define additional requirements on the Disable function.

3.2 Capability Group Compliance

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

4. IviScopeBase Capability Group

4.1 Overview

The IviScope base capabilities support oscilloscopes that can acquire waveforms from multiple channels with an edge trigger. The IviScope base capabilities define attributes and their values to configure the oscilloscope's channel, acquisition, and trigger sub-systems. The IviScope base capabilities also include functions for configuring the oscilloscope as well as initiating waveform acquisition and retrieving waveforms.

The IviScope base capabilities organize the configurable settings into three main categories: the channel sub-system, the acquisition sub-system, and the trigger sub-system.

4.1.1 Channel Sub-System

The channel sub-system configures the range of voltages the oscilloscope acquires and how the oscilloscope couples the input signal to the acquisition sub-system. The main channel sub-system attributes include:

- Channel Enabled
- Probe Attenuation
- Vertical Coupling
- Vertical Offset
- Vertical Range

All of the channel sub-system attributes represent a capability that is repeated on all instrument's channels. They can be set as a group with the Configure Channel function.

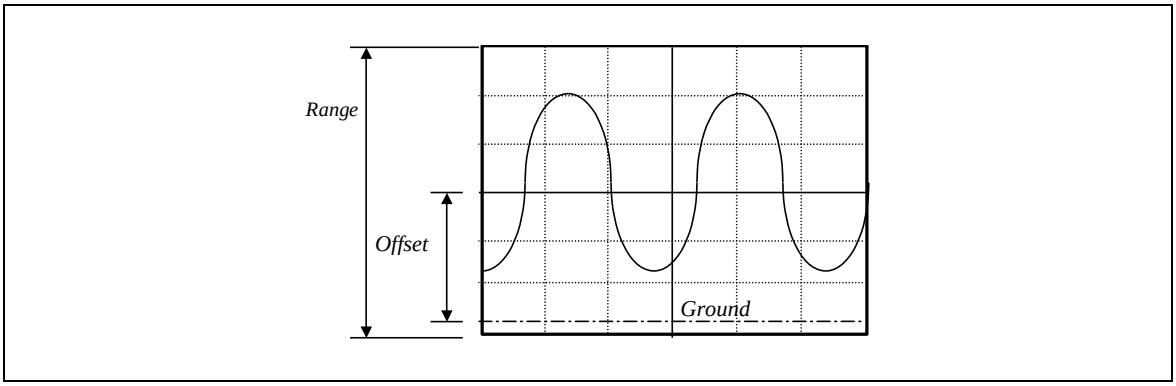


Figure 4-1. Channel Sub-System Attributes

The Vertical Range attribute specifies the absolute value of the range of voltages that the oscilloscope acquires. The Vertical Offset attribute specifies the center of the range specified by the Vertical Range attribute with respect to ground. The Vertical Coupling attribute specifies how to couple the input signal to the channel sub-system. The Probe Attenuation attribute specifies the scaling factor by which the probe attenuates the input signal. Typically, the value of the Probe Attenuation attribute determines the range values the driver accepts for the Vertical Range and Vertical Offset attributes. The Channel Enabled attribute specifies whether the oscilloscope acquires a waveform for the channel.

4.1.2 Acquisition Sub-System

The acquisition sub-system configures the acquisition type, the size of the waveform record, the length of time that corresponds to the overall waveform record, and the position of the first point in the waveform record relative to the Trigger Event. The configurable Acquisition sub-system attributes include:

- Acquisition Start Time
- Acquisition Type
- Horizontal Minimum Number of Points
- Horizontal Time Per Record

The end-user specifies how the oscilloscope acquires the data and fills the waveform record with the Acquisition Type attribute. The user specifies the minimum number of points they require the oscilloscope to acquire with the Horizontal Minimum Number of Points attribute. The Horizontal Time Per Record attribute specifies the length of time that corresponds to the overall waveform record. The Acquisition Start Time attribute specifies the position of the first point in the waveform record relative to the Trigger Event. If the value is positive, the first point in the waveform record occurs *after* the trigger event. If the value is negative, the first point in the waveform record occurs *before* the trigger event. The Figure 4-2 shows the effect of the negative acquisition start time value.

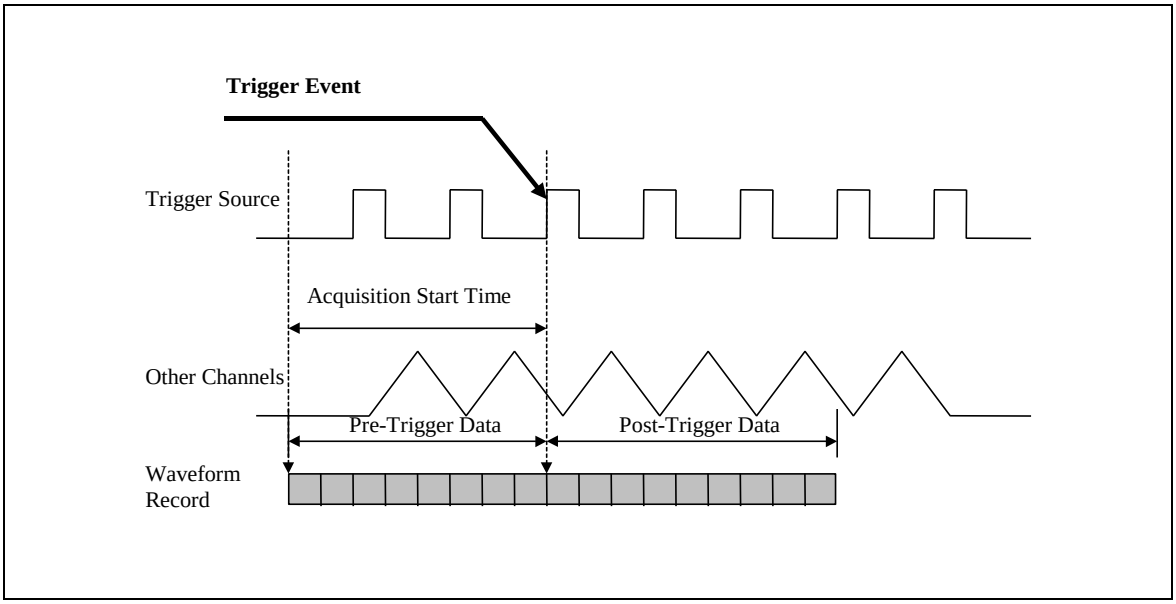


Figure 4-2. Acquisition Overview

Acquisition attributes can be set as a group with the Configure Acquisition Type (IVI-C only) and Configure Acquisition Record functions.

In addition, the acquisition sub-system includes two read-only attributes:

- Horizontal Record Length
- Horizontal Sample Rate

The Horizontal Record Length attribute returns the actual number of points in the waveform record. The Horizontal Sample Rate attribute returns the effective sample rate of the oscilloscope.

The IviScope class defines separate attributes for the minimum record size that the end-user requests and the actual record length. Typically, oscilloscopes change the record length dynamically when the acquisition type changes. For example, when the end-user changes the acquisition type from normal to

envelope, many oscilloscopes reduce the record length by half. When the end-user initiates a waveform acquisition, the instrument driver uses the value held in the Horizontal Minimum Number of Points attribute to check that the new record length is equal to or greater than the minimum record length the end-user requires.

4.1.3 Trigger Sub-System

The trigger sub-system configures the type of event that triggers the oscilloscope. The global trigger sub-system attributes are:

- Trigger Coupling
- Trigger Holdoff
- Trigger Type

The Trigger Type attribute specifies the event that triggers the oscilloscope.

The Trigger Holdoff attribute specifies the length of time after the oscilloscope detects a trigger during which the oscilloscope ignores additional triggers. The Trigger Holdoff attribute affects the instrument operation only when the oscilloscope requires multiple acquisitions to build a complete waveform. The oscilloscope requires multiple waveform acquisitions when the sample mode is equivalent time or the acquisition type is set to envelope or average.

The Trigger Coupling attribute specifies how the oscilloscope couples the trigger source to the trigger sub-system.

The attributes from the above list can be set as a group with the Configure Trigger and Configure Trigger Coupling (IVI-C only) functions.

The following attributes configure the edge trigger. These attributes can be set as a group with the Configure Edge Trigger Source function.

- Trigger Level
- Trigger Source
- Trigger Slope

The Trigger Level attribute specifies the voltage threshold for the trigger sub-system. The Trigger Source attribute specifies the source the oscilloscope monitors for the trigger event. Most of the trigger types use the values held in the Trigger Level and Trigger Source attributes.

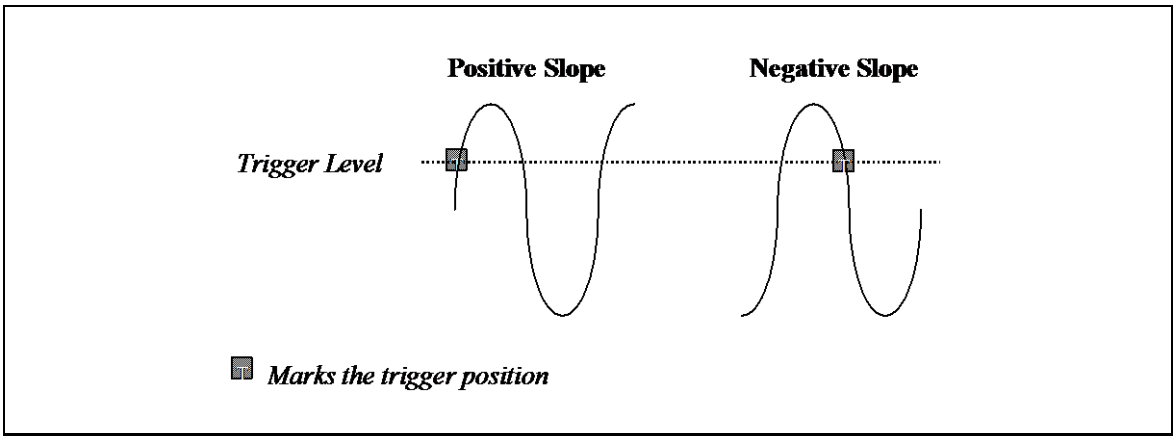


Figure 4-3. Edge Triggers

The Trigger Slope attribute specifies whether a positive or negative edge triggers the oscilloscope.

When the trigger type is edge, the values held in the Trigger Level, Trigger Source, and Trigger Slope attributes define the trigger event. The oscilloscope triggers when the signal from the trigger source crosses the threshold level with the polarity that the Trigger Level and Trigger Coupling attributes specify.

The following figure shows how the hold-off affects the trigger sub-system. Ideally the trigger event occurs at condition '1', but sometimes the oscilloscope triggers on condition '2' because the signal crosses the trigger level. When the end-user specifies an appropriate hold-off, the oscilloscope triggers on conditions '1' and '3', and ignores conditions '2' and '4'.

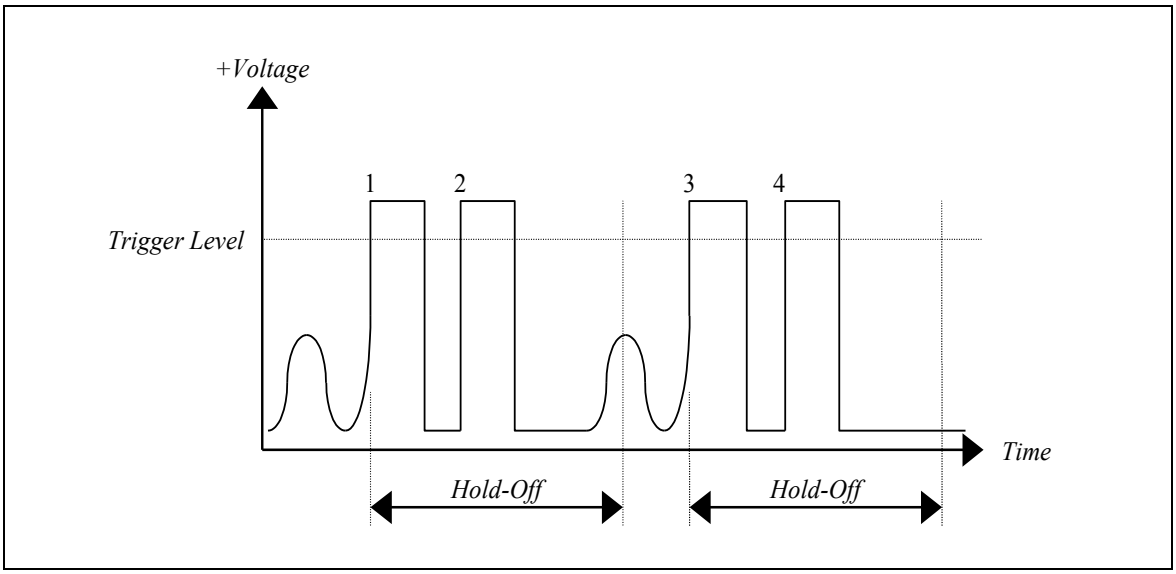


Figure 4-4. Trigger Hold-Off Overview

This IviScopeBase Capabilities define functions that retrieve waveforms from the oscilloscope. These functions return the following information:

- The waveform record as an array of voltages.
- The time that corresponds to the first point in the waveform array relative to the Trigger Event (acquisition start time).
- The effective time between points in the waveform record.

4.2 IviScopeBase Attributes

The IviScopeBase capability group defines the following attributes:

- Acquisition Start Time
- Acquisition Type
- Channel Count
- Channel Enabled
- Channel Item (IVI-COM only)
- Channel Name (IVI-COM only)
- Horizontal Minimum Number of Points
- Horizontal Record Length
- Horizontal Sample Rate
- Horizontal Time Per Record
- Input Impedance
- Maximum Input Frequency
- Measurement Channel Count (IVI-COM Only)
- Measurement Channel Item (IVI-COM Only)
- Probe Attenuation
- Trigger Coupling
- Trigger Holdoff
- Trigger Level
- Trigger Slope
- Trigger Source
- Trigger Type
- Vertical Coupling
- Vertical Offset
- Vertical Range

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

4.2.1 Acquisition Start Time

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

COM Property Name

Acquisition.StartTime

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_ACQUISITION_START_TIME

Description

Specifies the length of time from the trigger event to the first point in the waveform record. If the value is positive, the first point in the waveform record occurs *after* the trigger event. If the value is negative, the first point in the waveform record occurs *before* the trigger event. The units are seconds.

4.2.2 Acquisition Type

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Acquisition Type (IVI-C only)

COM Property Name

Acquisition.Type

COM Enumeration Name

IviScopeAcquisitionTypeEnum

C Constant Name

IVISCOPE_ATTR_ACQUISITION_TYPE

Description

Specifies how the oscilloscope acquires data and fills the waveform record.

Defined Values

Name	*Description*		
		Language	*Identifier*
Normal	Configures the oscilloscope to acquire one sample for each point in the waveform record. The oscilloscope uses real-time or equivalent time sampling.		
		C	IVISCOPE_VAL_NORMAL
		COM	IviScopeAcquisitionTypeNormal
High Resolution	Configures the oscilloscope to oversample the input signal. The oscilloscope calculates the average value that corresponds to each position in the waveform record. The oscilloscope uses only real-time sampling.		
		C	IVISCOPE_VAL_HI_RES
		COM	IviScopeAcquisitionTypeHiRes
Average	Configures the oscilloscope to acquire multiple waveforms and calculate the average value for each point in the waveform record. The end-user specifies the number of waveforms to acquire with the Number of Averages attribute. The oscilloscope uses real-time or equivalent time sampling.		
		C	IVISCOPE_VAL_AVERAGE
		COM	IviScopeAcquisitionTypeAverage
Peak Detect	Sets the oscilloscope to the peak-detect acquisition mode. The oscilloscope oversamples the input signal and keeps the minimum and maximum values that correspond to each position in the waveform record. The oscilloscope uses only real-time sampling.		
		C	IVISCOPE_VAL_PEAK_DETECT
		COM	IviScopeAcquisitionTypePeakDetect
Envelope	Sets the oscilloscope to the envelope acquisition mode. The oscilloscope acquires multiple waveforms and keeps the minimum and maximum voltages it acquires for each point in the waveform record. The end-user specifies the number of waveforms the oscilloscope acquires with the Number of Envelopes attribute. The oscilloscope can use real-time or equivalent-time sampling.		
		C	IVISCOPE_VAL_ENVELOPE
		COM	IviScopeAcquisitionTypeEnvelope

Compliance Notes

1. The driver shall implement the Normal value for this attribute.
2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_ACQUISITION_TYPE_CLASS_EXT_BASE and less than IVISCOPE_VAL_ACQUISITION_TYPE_SPECIFIC_EXT_BASE.
3. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_ACQUISITION_TYPE_SPECIFIC_EXT_BASE.
4. If an IVI Class-Compliant specific driver implements any of the defined values in the following table, it shall also implement the corresponding capability group:

Value	Required Capability Group
Peak Detect	IviScopeMinMaxWaveform
Envelope	IviScopeMinMaxWaveform
Average	IviScopeAverageAcquisition

5. 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 Acquisition Type Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of Acquisition Type Specific Ext Base, IVISCOPE_VAL_ACQUISITION_TYPE_SPECIFIC_EXT_BASE and IVISCOPE_VAL_ACQUISITION_TYPE_CLASS_EXT_BASE.

4.2.3 Channel Count

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

COM Property Name

Channels.Count

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_CHANNEL_COUNT

Description

Returns the number of available channels.

4.2.4 Channel Enabled

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

COM Property Name

Channels.Item().Enabled

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_CHANNEL_ENABLED

Description

Specifies whether the oscilloscope acquires a waveform for the channel.

Defined Values

Name	*Description*		
		Language	*Identifier*
True	The oscilloscope acquires a waveform for the channel		
		C	VI_TRUE
		COM	True
False	The oscilloscope does not acquire a waveform for the channel		
		C	VI_FALSE
		COM	False

Compliance Notes

Instrument drivers shall support the value True.

4.2.5 Channel Item (IVI-COM Only)

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

COM Property Name

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

COM Enumeration Name

N/A

C Constant Name

N/A

Description

Returns a pointer to the IIviScopeChannel interface.

4.2.6 Channel Name (IVI-COM Only)

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

COM Property Name

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

and

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

COM Enumeration Name

N/A

C Constant Name

N/A.

(Use the `GetChannelName` function.)

Description

Returns the physical repeated capability identifier defined by the specific driver for the channel that corresponds to the one-based index that the user specifies. If the driver defines a qualified channel name, this property returns the qualified name.

Valid values for the `ChannelIndex` parameter are between one and the value of the Channel Count attribute. If the user passes an invalid value for the `ChannelIndex` parameter, the value of this attribute is an empty string.

4.2.7 Horizontal Minimum Number of Points

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

COM Property Name

Acquisition.NumberOfPointsMin

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_HORZ_MIN_NUM_PTS

Description

Specifies the minimum number of points the end-user requires in the waveform record for each channel. The instrument driver uses the value the end-user specifies to configure the record length that the oscilloscope uses for waveform acquisition. If the instrument cannot support the requested record length, the driver shall configure the instrument to the closest bigger record length. The Horizontal Record Length attribute returns the actual record length.

4.2.8 Horizontal Record Length

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	RO	N/A	N/A	Actual Record Length (IVI-C only)

COM Property Name

Acquisition.RecordLength

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_HORZ_RECORD_LENGTH

Description

Returns the actual number of points the oscilloscope acquires for each channel. The value is equal to or greater than the minimum number of points the end-user specifies with the Horizontal Minimum Number of Points attribute.

Note: Oscilloscopes may use different size records depending on the value the user specifies for the Acquisition Type attribute.

4.2.9 Horizontal Sample Rate

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	N/A	N/A	Sample Rate (IVI-C only)

COM Property Name

Acquisition.SampleRate

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_HORZ_SAMPLE_RATE

Description

Returns the effective sample rate of the acquired waveform using the current configuration. The units are samples per second.

4.2.10 Horizontal Time Per Record

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	N/A	Up	Configure Acquisition Record

COM Property Name

Acquisition.TimePerRecord

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_HORZ_TIME_PER_RECORD

Description

Specifies the length of time that corresponds to the record length. The units are seconds.

4.2.11 Input Impedance

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	Channels	None	Configure Channel Characteristics

COM Property Name

Channels.Item().InputImpedance

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_INPUT_IMPEDANCE

Description

Specifies the input impedance for the channel in Ohms.

Common values are 50.0, 75.0, and 1,000,000.0.

4.2.12 Maximum Input Frequency

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	Channels	Up	Configure Channel Characteristics

COM Property Name

Channels.Item().InputFrequencyMax

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_MAX_INPUT_FREQUENCY

Description

Specifies the maximum frequency for the input signal you want the instrument to accommodate without attenuating it by more than 3dB. If the bandwidth limit frequency of the instrument is greater than this maximum frequency, the driver enables the bandwidth limit. This attenuates the input signal by at least 3dB at frequencies greater than the bandwidth limit.

4.2.13 Measurement Channel Count (IVI-COM Only)

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

COM Property Name

Measurements.Count

COM Enumeration Name

N/A

C Constant Name

N/A

Description

Returns the number of available measurement channels.

4.2.14 Measurement Channel Item (IVI-COM Only)

Data Type	Access	Applies to	Coercion	High Level Functions
IIviScopeMeasurement*	RO	Measurements	None	None

COM Property Name

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

COM Enumeration Name

N/A

C Constant Name

N/A

Description

Returns a pointer to the IIviScopeMeasurement interface.

4.2.15 Probe Attenuation

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	Channels	None	Configure Channel

COM Property Name

Channels.Item().ProbeAttenuation

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_PROBE_ATTENUATION

Description

Specifies the scaling factor by which the probe the end-user attaches to the channel attenuates the input. If the oscilloscope is auto sensing the probe attenuation, setting this attribute to a positive value configures the oscilloscope to use the manual probe attenuation the end-user specifies. For example, for a 10:1 probe, the end-user sets this attribute to 10.0.

After this attribute is set to a positive value, the application programs should set this attribute to reflect the probe attenuation each time the probe physically changes.

This specification reserves negative values to control oscilloscopes with auto probe sense capability. The *IviScopeProbeAutoSense* extension section defines values for controlling the auto probe sense.

Defined Values

Name	***Description***		
		Language	***Identifier***
Probe Sense On	Configures the oscilloscope to sense the attenuation of the probe automatically. After the end-user enables the automatic probe sense, subsequent queries of this attribute return the value Probe Sense On. Use the Probe Sense Value attribute to obtain the actual probe attenuation. If the oscilloscope is set to sense the probe attenuation automatically, setting this attribute to a value different from Probe Sense On disables the automatic probe sense and configures the oscilloscope to use the manual probe attenuation the end-user specifies.		
		C	IVISCOPE_VAL_PROBE_SENSE_ON
		COM	-1.0

Compliance Notes

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

3. If an IVI Class Compliant specific driver implements any of the defined values in the following table, it shall also implement the corresponding capability group:

Value	Required Capability Group
Probe Sense On	IviScopeProbeAutoSense

4. 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 Probe Attenuation Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of Probe Attenuation Specific Ext Base, `IVISCOPE_VAL_PROBE_ATTENUATION_SPECIFIC_EXT_BASE` and `IVISCOPE_VAL_PROBE_ATTENUATION_CLASS_EXT_BASE`.

4.2.16 Trigger Coupling

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Trigger Coupling (IVI-C only)

COM Property Name

Trigger.Coupling

COM Enumeration Name

IviScopeTriggerCouplingEnum

C Constant Name

IVISCOPE_ATTR_TRIGGER_COUPLING

Description

Specifies how the oscilloscope couples the trigger source.

Defined Values

Name	*Description*		
		Language	*Identifier*
AC	The oscilloscope AC couples the trigger signal.		
		C	IVISCOPE_VAL_AC
		COM	IviScopeTriggerCouplingAC
DC	The oscilloscope DC couples the trigger signal.		
		C	IVISCOPE_VAL_DC
		COM	IviScopeTriggerCouplingDC
LF Reject	The oscilloscope filters out the low frequencies from the trigger signal.		
		C	IVISCOPE_VAL_LF_REJECT
		COM	IviScopeTriggerCouplingLFReject
HF Reject	The oscilloscope filters out the high frequencies from the trigger signal.		
		C	IVISCOPE_VAL_HF_REJECT
		COM	IviScopeTriggerCouplingHFReject
Noise Reject	The oscilloscope filters out the noise from the trigger signal.		
		C	IVISCOPE_VAL_NOISE_REJECT
		COM	IviScopeTriggerCouplingNoiseReject

Compliance Notes

1. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TRIGGER_COUPLING_CLASS_EXT_BASE and less than IVISCOPE_VAL_TRIGGER_COUPLING_SPECIFIC_EXT_BASE.
2. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TRIGGER_COUPLING_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 Coupling Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of Trigger Coupling Specific Ext Base, `IVISCOPE_VAL_TRIGGER_COUPLING_SPECIFIC_EXT_BASE` and `IVISCOPE_VAL_TRIGGER_COUPLING_CLASS_EXT_BASE`.

4.2.17 Trigger Holdoff

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	N/A	Note[1]	Configure Trigger

COM Property Name

Trigger.Holdoff

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_TRIGGER_HOLDOFF

Description

Specifies the length of time the oscilloscope waits after it detects a trigger until the oscilloscope enables the trigger subsystem to detect another trigger. The units are seconds. The Trigger Holdoff attribute affects instrument operation only when the oscilloscope requires multiple acquisitions to build a complete waveform. The oscilloscope requires multiple waveform acquisitions when it uses equivalent-time sampling or when the Acquisition Type attribute is set to Envelope or Average.

Note: Many scopes have a small, non-zero value as the minimum value for this attribute. To configure the instrument to use the shortest trigger hold-off, the user can specify a value of zero for this attribute. Therefore, the IVI Class-Compliant specific driver shall coerce any value between zero and the minimum value to the minimum value. No other coercion is allowed on this attribute.

4.2.18 Trigger Level

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	N/A	None	Configure Edge Trigger Source Configure Glitch Trigger Source Configure Width Trigger Source

COM Property Name

Trigger.Level

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_TRIGGER_LEVEL

Description

Specifies the voltage threshold for the trigger sub-system. The units are volts. This attribute affects instrument behavior only when the Trigger Type is set to one of the following values: Edge Trigger, Glitch Trigger, or Width Trigger.

This attribute, along with the Trigger Slope, Trigger Source, and Trigger Coupling attributes, defines the trigger event when the Trigger Type is set to Edge Trigger.

4.2.19 Trigger Slope

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

COM Property Name

Trigger.Edge.Slope

COM Enumeration Name

IviScopeTriggerSlopeEnum

C Constant Name

IVISCOPE_ATTR_TRIGGER_SLOPE

Description

Specifies whether a rising or a falling edge triggers the oscilloscope.

This attribute affects instrument operation only when the Trigger Type attribute is set to Edge Trigger.

Defined Values

Name	*Description*		
		Language	*Identifier*
Positive	A positive (rising) edge passing through the trigger level triggers the oscilloscope.		
		C	IVISCOPE_VAL_POSITIVE
		COM	IviScopeTriggerSlopePositive
Negative	A negative (falling) edge passing through the trigger level triggers the oscilloscope.		
		C	IVISCOPE_VAL_NEGATIVE
		COM	IviScopeTriggerSlopeNegative

4.2.20 Trigger Source

Data Type	Access	Applies to	Coercion	High Level Functions
ViString	R/W	N/A	None	Configure Edge Trigger Source Configure TV Trigger Source Configure Runt Trigger Source Configure Glitch Trigger Source Configure Width Trigger Source

COM Property Name

Trigger.Source

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_TRIGGER_SOURCE

Description

Specifies the source the oscilloscope monitors for the trigger event. The value can be a channel name alias, a driver-specific channel string, or one of the values below.

This attribute affects the instrument operation only when the Trigger Type is set to one of the following values: Edge Trigger, TV Trigger, Runt Trigger, Glitch Trigger, or Width Trigger .

Defined Values:

Name	*Description*		
		Language	*Identifier*
External	The oscilloscope waits for a trigger on the external trigger input.		
		C	IVISCOPE_VAL_EXTERNAL
		COM	"VAL_EXTERNAL"
TTL0	The oscilloscope waits until it receives a trigger on the TTL0 line.		
		C	IVISCOPE_VAL_TTL0
		COM	"VAL_TTL0"
TTL1	The oscilloscope waits until it receives a trigger on the TTL1 line.		
		C	IVISCOPE_VAL_TTL1
		COM	"VAL_TTL1"
TTL2	The oscilloscope waits until it receives a trigger on the TTL2 line.		
		C	IVISCOPE_VAL_TTL2
		COM	"VAL_TTL2"
TTL3	The oscilloscope waits until it receives a trigger on the TTL3 line.		
		C	IVISCOPE_VAL_TTL3
		COM	"VAL_TTL3"
TTL4	The oscilloscope waits until it receives a trigger on the TTL4 line.		
		C	IVISCOPE_VAL_TTL4
		COM	"VAL_TTL4"
TTL5	The oscilloscope waits until it receives a trigger on the TTL5 line.		
		C	IVISCOPE_VAL_TTL5
		COM	"VAL_TTL5"
TTL6	The oscilloscope waits until it receives a trigger on the TTL6 line.		
		C	IVISCOPE_VAL_TTL6
		COM	"VAL_TTL6"
TTL7	The oscilloscope waits until it receives a trigger on the TTL7 line.		
		C	IVISCOPE_VAL_TTL7
		COM	"VAL_TTL7"
ECL0	The oscilloscope waits until it receives a trigger on the ECL0 line.		
		C	IVISCOPE_VAL_ECL0
		COM	"VAL_ECL0"
ECL1	The oscilloscope waits until it receives a trigger on the ECL1 line.		
		C	IVISCOPE_VAL_ECL1
		COM	"VAL_ECL1"
PXI Star	The oscilloscope waits until it receives a trigger on the PXI Star bus.		
		C	IVISCOPE_VAL_PXI_STAR
		COM	"VAL_PXI_STAR"

Name	*Description*		
		Language	*Identifier*
RTSI0	The oscilloscope waits until it receives a trigger on the RTSI0 line.		
		C	IVISCOPE_VAL_RTSI_0
		COM	"VAL_RTSI_0"
RTSI1	The oscilloscope waits until it receives a trigger on the RTSI1 line.		
		C	IVISCOPE_VAL_RTSI_1
		COM	"VAL_RTSI_1"
RTSI2	The oscilloscope waits until it receives a trigger on the RTSI2 line.		
		C	IVISCOPE_VAL_RTSI_2
		COM	"VAL_RTSI_2"
RTSI3	The oscilloscope waits until it receives a trigger on the RTSI3 line.		
		C	IVISCOPE_VAL_RTSI_3
		COM	"VAL_RTSI_3"
RTSI4	The oscilloscope waits until it receives a trigger on the RTSI4 line.		
		C	IVISCOPE_VAL_RTSI_4
		COM	"VAL_RTSI_4"
RTSI5	The oscilloscope waits until it receives a trigger on the RTSI5 line.		
		C	IVISCOPE_VAL_RTSI_5
		COM	"VAL_RTSI_5"
RTSI6	The oscilloscope waits until it receives a trigger on the RTSI6 line.		
		C	IVISCOPE_VAL_RTSI_6
		COM	"VAL_RTSI_6"

4.2.21 Trigger Type

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

COM Property Name

Trigger.Type

COM Enumeration Name

IviScopeTriggerTypeEnum

C Constant Name

IVISCOPE_ATTR_TRIGGER_TYPE

Description

Specifies the event that triggers the oscilloscope.

Defined Values

Name	*Description*		
		Language	*Identifier*
Edge Trigger	Configures the oscilloscope for edge triggering. An edge trigger occurs when the trigger signal specified with the Trigger Source attribute passes the voltage threshold specified with the Trigger Level attribute and has the slope specified with the Trigger Slope attribute.		
		C	IVISCOPE_VAL_EDGE_TRIGGER
		COM	IviScopeTriggerEdge
TV Trigger	Configures the oscilloscope for triggering on TV signals. Use the IviScopeTVTrigger extension attributes and functions to configure the trigger.		
		C	IVISCOPE_VAL_TV_TRIGGER
		COM	IviScopeTriggerTV
Runt Trigger	Configures the oscilloscope for runt triggering. Use the IviScopeRuntTrigger extension attributes and functions to configure the trigger.		
		C	IVISCOPE_VAL_RUNT_TRIGGER
		COM	IviScopeTriggerRunt
Glitch Trigger	Configures the oscilloscope for glitch triggering. Use the IviScopeGlitchTrigger extension attributes and functions to configure the trigger.		
		C	IVISCOPE_VAL_GLITCH_TRIGGER
		COM	IviScopeTriggerGlitch
Width Trigger	Configures the oscilloscope for width triggering. Use the IviScopeWidthTrigger extension attributes and functions to configure the trigger.		
		C	IVISCOPE_VAL_WIDTH_TRIGGER
		COM	IviScopeTriggerWidth
Immediate Trigger	Configures the oscilloscope for immediate triggering. The oscilloscope does not wait for trigger of any kind upon initialization.		

		C	IVISCOPE_VAL_IMMEDIATE_TRIGGER
		COM	IviScopeTriggerImmediate
AC Line Trigger	Configures the oscilloscope for AC Line triggering. Use the IviScopeACLineTrigger extension attributes and functions to configure the trigger.		
		C	IVISCOPE_VAL_AC_LINE_TRIGGER
		COM	IviScopeTriggerACLine

Compliance Notes

1. The IVI Class-Compliant specific driver shall implement the Edge Trigger value for this attribute.
2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TRIGGER_TYPE_CLASS_EXT_BASE and less than IVISCOPE_VAL_TRIGGER_TYPE_SPECIFIC_EXT_BASE.
3. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TRIGGER_TYPE_SPECIFIC_EXT_BASE.
4. If an IVI Class-Compliant specific driver implements any of the defined values in the following table, it shall also implement the corresponding capability group:

Value	Required Capability Group
TV Trigger	IviScopeTVTrigger
Runt Trigger	IviScopeRuntTrigger
Glitch Trigger	IviScopeGlitchTrigger
Width Trigger	IviScopeWidthTrigger
AC Line Trigger	IviScopeAcLineTrigger

5. 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 Type Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of Trigger Type Specific Ext Base, IVISCOPE_VAL_TRIGGER_TYPE_SPECIFIC_EXT_BASE and IVISCOPE_VAL_TRIGGER_TYPE_CLASS_EXT_BASE.

4.2.22 Vertical Coupling

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	Channels	None	Configure Channel

COM Property Name

Channels.Item().Coupling

COM Enumeration Name

IviScopeVerticalCouplingEnum

C Constant Name

IVISCOPE_ATTR_VERTICAL_COUPLING

Description

Specifies how the oscilloscope couples the input signal for the channel.

Defined Values

Name	*Description*		
		Language	*Identifier*
AC	The oscilloscope AC couples the input signal.		
		C	IVISCOPE_VAL_AC
		COM	IviScopeVerticalCouplingAC
DC	The oscilloscope DC couples the input signal.		
		C	IVISCOPE_VAL_DC
		COM	IviScopeVerticalCouplingDC
Gnd	The oscilloscope couples the channel to the ground.		
		C	IVISCOPE_VAL_GND
		COM	IviScopeVerticalCouplingGnd

Compliance Notes

1. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_VERTICAL_COUPLING_CLASS_EXT_BASE and less than IVISCOPE_VAL_VERTICAL_COUPLING_SPECIFIC_EXT_BASE.
2. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_VERTICAL_COUPLING_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 Vertical Coupling Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of Vertical Coupling Specific Ext Base, IVISCOPE_VAL_VERTICAL_COUPLING_SPECIFIC_EXT_BASE and IVISCOPE_VAL_VERTICAL_COUPLING_CLASS_EXT_BASE.

4.2.23 Vertical Offset

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	Channels	None	Configure Channel

COM Property Name

Channels.Item().Offset

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_VERTICAL_OFFSET

Description

Specifies the location of the center of the range that the Vertical Range attribute specifies. The value is with respect to ground and is in volts.

For example, to acquire a sine wave that spans between on 0.0 and 10.0 volts, set this attribute to 5.0 volts.

4.2.24 Vertical Range

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	R/W	Channel	Up	Configure Channel

COM Property Name

Channels.Item().Range

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_VERTICAL_RANGE

Description

Specifies the absolute value of the full-scale input range for a channel. The units are volts.

For example, to acquire a sine wave that spans between -5.0 and 5.0 volts, set the Vertical Range attribute to 10.0 volts.

4.3 IviScopeBase Functions

The IviScopeBase capability group defines the following functions:

- Abort
- Acquisition Status
- Actual Record Length (IVI-C only)
- Configure Acquisition Record
- Configure Acquisition Type (IVI-C only)
- Configure Channel
- Configure Channel Characteristics
- Configure Edge Trigger Source
- Configure Trigger
- Configure Trigger Coupling (IVI-C only)
- GetChannelName (IVI-C Only)
- Fetch Waveform
- Initiate Acquisition
- Is Invalid Waveform Element
- Read Waveform
- Sample Rate (IVI-C only)

This section describes the behavior and requirements of each function.

4.3.1 Abort

Description

This function aborts an acquisition and returns the oscilloscope to the Idle state. This function does not check the instrument status.

Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

If the instrument cannot abort an initiated acquisition, the driver shall return the Function Not Supported error.

COM Method Prototype

```
HRESULT Measurements.Abort ();
```

C Prototype

```
ViStatus IviScope_Abort (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Return Values

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

4.3.2 Acquisition Status

Description

This function returns whether an acquisition is in progress, complete, or if the status is unknown.

Note: This function returns in the `status` output parameter only the values defined below.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

If the driver cannot query the instrument to determine its state, the driver returns the Acquisition Status Unknown value.

COM Enumeration Name

```
IviScopeAcquisitionStatusEnum
```

COM Method Prototype

```
HRESULT Measurements.Status ([out,retval] IviScopeAcquisitionStatusEnum *Status);
```

C Prototype

```
ViStatus IviScope_AcquisitionStatus (ViSession Vi,
                                     ViInt32 *Status);
```

Parameters

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

Outputs	Description	Base Type
`Status`	Acquisition Status	`ViInt32`

Defined Values for Status Parameter

Name	***Description***		
		Language	***Identifier***
Acquisition Complete	The oscilloscope has completed the acquisition.		
		C	`IVISCOPE_VAL_ACQ_COMPLETE`
		COM	`IviScopeAcquisitionStatusComplete`
Acquisition In Progress	The oscilloscope is still acquiring data.		
		C	`IVISCOPE_VAL_ACQ_IN_PROGRESS`
		COM	`IviScopeAcquisitionStatusInProgress`
Acquisition Status Unknown	The oscilloscope cannot determine the status of the acquisition.		
		C	`IVISCOPE_VAL_ACQ_STATUS_UNKNOWN`
		COM	`IviScopeAcquisitionStatusUnknown`

Return Values

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

4.3.3 Actual Record Length (IVI-C only)

Description

This function returns the actual number of points the oscilloscope acquires per channel. After configuring the oscilloscope for an acquisition, call this function to determine the size of the waveforms that the oscilloscope acquires. The value is equal to or greater than the minimum number of points specified in the Configure Acquisition Record function.

For IVI-C drivers, allocate a `ViReal64` array of this size or greater to pass as the `WaveformArray` parameter of the `IviScope_ReadWaveform` and `IviScope_FetchWaveform` functions.

The oscilloscope may use different size records depending on the acquisition type. Specify the acquisition type with the Configure Acquisition Type function.

COM Method Prototype

`N/A`
(use the `Acquisition.RecordLength` property)

C Prototype

```
ViStatus IviScope_ActualRecordLength (ViSession Vi,
                                      ViInt32 *ActualRecordLength);
```

Parameters

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

Outputs	Description	Base Type
`ActualRecord Length`	Returns the actual number of points the oscilloscope acquires for each channel. The driver returns the value held in the Horizontal Record Length attribute. See the Horizontal Record Length attribute for a complete description.	`ViInt32`

Return Values

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

4.3.4 Configure Acquisition Record

Description

This function configures the most commonly configured attributes of the oscilloscope acquisition sub-system. These attributes are the time per record, minimum record length, and the acquisition start time.

COM Method Prototype

```
HRESULT Acquisition.ConfigureRecord ([in] DOUBLE TimePerRecord,
                                     [in] LONG MinNumPts,
                                     [in] DOUBLE AquisitionStartTime);
```

C Prototype

```
ViStatus IviScope_ConfigureAcquisitionRecord (ViSession Vi,
                                              ViReal64 TimePerRecord,
                                              ViInt32 MinNumPts,
                                              ViReal64 AcquisitionStartTime);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
TimePerRecord	Specifies the time per record. The driver uses this value to set the Horizontal Time Per Record attribute. See the attribute description for more information.	ViReal64
MinNumPts	Specifies the minimum number of points the end-user allows in the waveform recorded. The driver uses this value to set the Horizontal Minimum Number of Points attribute. See the attribute description for more information.	ViInt32
AcquisitionStartTime	Specifies the position of the first point in the waveform record relative to the trigger event. The driver uses this value to set the Acquisition Start Time attribute. See the attribute description for more information.	ViReal64

Return Values

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

4.3.5 Configure Acquisition Type (IVI-C only)

Description

This function configures how the oscilloscope acquires data and fills the waveform record.

COM Method Prototype

`N/A`
(use the `Acquisition.Type` property)

C Prototype

```
ViStatus IviScope_ConfigureAcquisitionType (ViSession Vi,
                                            ViInt32 AcquisitionType);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`AcquisitionType`	Specifies the manner in which the oscilloscope acquires data and fills the waveform record. The driver sets the Acquisition Type attribute to this value. See the attribute description for more information.	`ViInt32`

Return Values

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

4.3.6 Configure Channel

Description

This function configures the most commonly configured attributes of the oscilloscope channel sub-system. These attributes are the range, offset, coupling, probe attenuation, and whether the channel is enabled.

COM Method Prototype

```
HRESULT Channels.Item().Configure ([in] DOUBLE Range,
                                   [in] DOUBLE Offset,
                                   [in] IviScopeVerticalCouplingEnum Coupling,
                                   [in] DOUBLE ProbeAttenuation,
                                   [in] VARIANT_BOOL Enabled);
```

C Prototype

```
ViStatus IviScope_ConfigureChannel (ViSession Vi,
                                    ViConstString Channel,
                                    ViReal64 Range,
                                    ViReal64 Offset,
                                    ViInt32 Coupling,
                                    ViReal64 ProbeAttenuation,
                                    ViBoolean Enabled);
```

Parameters

Inputs	Description	Data Type
Vi	Instrument handle	ViSession
Channel	The name of the oscilloscope channel to configure.	ViConstString
Range	Specifies the vertical range. The driver uses this value to set the Vertical Range attribute. See the attribute description for more information.	ViReal64
Offset	Specifies the vertical offset. The driver uses this value to set the Vertical Offset attribute. See the attribute description for more information.	ViReal64
Coupling	Specifies how to couple the input signal. The driver uses this value to set the Vertical Coupling attribute. See the attribute description for more information.	ViInt32
ProbeAttenuation	Specifies the probe attenuation. The driver uses this value to set the Probe Attenuation attribute. See the attribute description for more information	ViReal64
Enabled	Specifies if the channel is enabled for acquisition. The driver uses this value to set the Channel Enabled attribute. See the attribute description for more information	ViBoolean

Return Values

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

4.3.7 Configure Channel Characteristics

Description

This function configures the attributes that control the electrical characteristics of the channel. These attributes are the input impedance and the maximum frequency of the input signal.

COM Method Prototype

```
HRESULT Channels.Item().ConfigureCharacteristics ([in] DOUBLE InputImpedance,
                                                  [in] DOUBLE MaxInputFrequency);
```

C Prototype

```
ViStatus IviScope_ConfigureChanCharacteristics (ViSession Vi,
                                                ViConstString Channel,
                                                ViReal64 InputImpedance,
                                                ViReal64 MaxInputFrequency);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Channel	Name of the channel to configure.	ViConstString
InputImpedance	The input impedance for the channel. The driver sets the Input Impedance to this value. See the Input Impedance attribute for a complete description and defined values.	ViReal64
MaxInputFrequency	The maximum input frequency for the channel. The driver sets the Max Input Frequency to this value. See the attribute description for more information.	ViReal64

Return Values

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

4.3.8 Configure Edge Trigger Source

Description

This function sets the edge triggering attributes. An edge trigger occurs when the trigger signal that the end-user specifies with the `Source` parameter passes through the voltage threshold that the end-user specifies with the `level` parameter and has the slope that the end-user specifies with the `Slope` parameter.

This function affects instrument behavior only if the Trigger Type is Edge Trigger. Set the Trigger Type and Trigger Coupling before calling this function.

If the trigger source is one of the analog input channels, an application program should configure the vertical range, vertical offset, vertical coupling, probe attenuation, and the maximum input frequency before calling this function.

COM Method Prototype

```
HRESULT Trigger.Edge.Configure ([in] BSTR Source,
                                [in] DOUBLE Level,
                                [in] IviScopeTriggerSlopeEnum Slope);
```

C Prototype

```
ViStatus IviScope_ConfigureEdgeTriggerSource (ViSession Vi,
                                              ViConstString Source,
                                              ViReal64 Level,
                                              ViInt32 Slope);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Source`	Specifies the trigger source. The driver uses this value to set the Trigger Source attribute. See the attribute description for more information.	`ViConstString`
`Level`	Specifies the trigger level. The driver uses this value to set the Trigger Level attribute. See the attribute description for more information.	`ViReal64`
`Slope`	Specifies the trigger slope. The driver uses this value to set the Trigger Slope attribute. See the attribute description for more information.	`ViInt32`

Return Values

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

4.3.9 ConfigureTrigger

Description

This function configures the common attributes of the trigger subsystem. These attributes are the trigger type and trigger holdoff.

When the end-user calls Read Waveform, Read Waveform Measurement, Read Min Max Waveform, or Initiate Acquisition, the oscilloscope waits for a trigger. The end-user specifies the type of trigger for which the oscilloscope waits with the `TriggerType` parameter.

If the oscilloscope requires multiple waveform acquisitions to build a complete waveform, it waits for the length of time the end-user specifies with the `Holdoff` parameter to elapse since the previous trigger. The oscilloscope then waits for the next trigger. Once the oscilloscope acquires a complete waveform, it returns to the idle state.

COM Method Prototype

```
HRESULT Trigger.Configure ([in] IviScopeTriggerTypeEnum Type,
                           [in] DOUBLE Holdoff);
```

C Prototype

```
ViStatus IviScope_ConfigureTrigger (ViSession Vi,
                                    ViInt32 TriggerType,
                                    ViReal64 Holdoff);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`TriggerType`	Specifies the trigger type. The driver uses this value to set the Trigger Type attribute. See the attribute description for more information.	`ViInt32`
`Holdoff`	Specifies the trigger hold-off. The driver uses this value to set the Trigger Holdoff attribute. See the attribute description for more information.	`ViReal64`

Return Values

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

4.3.10 Configure Trigger Coupling (IVI-C only)

Description

This function sets the trigger coupling attribute.

COM Method Prototype

`N/A`
(use the `Trigger.Coupling` property)

C Prototype

```
ViStatus IviScope_ConfigureTriggerCoupling (ViSession Vi,
                                            ViInt32 Coupling);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Coupling`	Specifies the trigger coupling. The driver uses this value to set the Trigger Coupling attribute. See the attribute description for more information.	`ViInt32`

Return Values

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

4.3.11 Get Channel Name (IVI-C Only)

Description

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

COM Method Prototype

N/A. Use the `Channels.Name` and `Measurements.Name` properties.

C Prototype

```
ViStatus IviScope_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`
`NameBufferSize`	The number of bytes in the ViChar array that the user specifies for the `Name` parameter.	`ViInt32`

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

Return Values

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

4.3.12 Fetch Waveform

Description

This function returns the waveform the oscilloscope acquires for the specified channel. The waveform is from a previously initiated acquisition.

You use the Initiate Acquisition function to start an acquisition on the channels that the end-user configures with the Configure Channel function. The oscilloscope acquires waveforms on the concurrently enabled channels. If the channel is not enabled for the acquisition, this function returns the Channel Not Enabled error.

Use this function only when the acquisition mode is Normal, Hi Res, or Average. If the acquisition type is not one of the listed types, the function returns the Invalid Acquisition Type error.

You use the Acquisition Status function to determine when the acquisition is complete. You must call this function separately for each enabled channel to obtain the waveforms.

You can call the Read Waveform function instead of the Initiate Acquisition function. The Read Waveform function starts an acquisition on all enabled channels, waits for the acquisition to complete, and returns the waveform for the specified channel. You call this function to obtain the waveforms for each of the remaining channels.

After this function executes, each element in the `waveformArray` parameter is either a voltage or a value indicating that the oscilloscope could not sample a voltage.

The C end-user configures the interpolation method the oscilloscope uses with the `IviScope_ConfigureInterpolation` function. The COM end-user uses the `Acquisition.Interpolation` property. If interpolation is disabled, the oscilloscope does not interpolate points in the waveform. If the oscilloscope cannot sample a value for a point in the waveform, the driver sets the corresponding element in the `waveformArray` to an IEEE-defined NaN (Not a Number) value and the function returns the Invalid Waveform Element warning.

Use the Is Waveform Element Invalid function to test each element in the `waveformArray` parameter for an invalid waveform element.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

COM Method Prototype

```
HRESULT Measurements.Item().FetchWaveform (
                                    [in,out] SAFEARRAY(DOUBLE) *WaveformArray,
                                    [in,out] DOUBLE *InitialX,
                                    [in,out] DOUBLE *XIncrement);
```

C Prototype

```
ViStatus IviScope_FetchWaveform (ViSession Vi,
                                 ViConstString Channel,
                                 ViInt32 WaveformSize,
                                 ViReal64 WaveformArray[],
                                 ViInt32 *ActualPoints,
                                 ViReal64 *InitialX,
                                 ViReal64 *XIncrement);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Channel	Name of the channel from which to fetch a waveform.	ViConstString
WaveformSize	Specifies the number of elements in the waveform array.	ViInt32

Outputs	Description	Base Type
WaveformArray	A user-allocated (for IVI-C) or driver-allocated (for IVI-COM) buffer into which the acquired waveform is stored. Units for the individual array elements are in volts.	ViReal64[]
ActualPoints	Number of points actually placed in the waveform array.	ViInt32
InitialX	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	ViReal64
XIncrement	The effective time between points in the acquired waveform in seconds.	ViReal64

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 Waveform Element	Warning: One Of The Elements In The Waveform Array Is Invalid.
Invalid Acquisition Type	Error: Invalid Acquisition Type
Channel Not Enabled	Error: Specified Channel Is Not Enabled For Acquisition.

4.3.13 Initiate Acquisition

Description

This function initiates a waveform acquisition. After calling this function, the oscilloscope leaves the idle state and waits for a trigger. The oscilloscope acquires a waveform for each channel the end-user has enabled with the Configure Channel function.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

COM Method Prototype

```
HRESULT Measurements.Initiate();
```

C Prototype

```
ViStatus IviScope_InitiateAcquisition (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Return Values

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

4.3.14 Is Waveform Element Invalid

Description

This function determines whether a value you pass from the waveform array is invalid. After the read and fetch waveform functions execute, each element in the waveform array contains either a voltage or a value indicating that the oscilloscope could not sample a voltage. The driver uses an IEEE-defined NaN (Not a Number) value to mark as invalid each element in the waveform array for which the oscilloscope could not sample a voltage. This function determines whether a value you pass from the waveform array is invalid. Refer to *IVI-3.12: Floating Point Services Specification* for more information on how to return standard floating point values to user programs.

COM Method Prototype

```
HRESULT Measurements.IsWaveformElementInvalid ([in] DOUBLE Element,
                                               [out, retval] VARIANT_BOOL *IsInvalid);
```

C Prototype

```
ViStatus IviScope_IsInvalidWfmElement (ViSession Vi,
                                       ViReal64 ElementValue,
                                       ViBoolean *IsInvalid);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
ElementValue	Pass one of the values from the waveform array returned by the read and fetch waveform functions.	ViReal64

Outputs	Description	Base Type
IsInvalid	Returns whether the elementValue is a valid voltage or a value indicating that the oscilloscope could not sample a voltage. Valid return values: True - The elementValue indicates that the oscilloscope could not sample the voltage. False - The elementValue is a valid voltage.	ViBoolean

Return Values

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

4.3.15 Read Waveform

Description

This function initiates an acquisition on the channels that the end-user configures with the Configure Channel function. If the channel is not enabled for the acquisition, this function returns Channel Not Enabled error. It then waits for the acquisition to complete, and returns the waveform for the channel the end-user specifies. If the oscilloscope did not complete the acquisition within the time period the user specified with the `MaxTimeMilliseconds` parameter, the function returns the Max Time Exceeded error.

Use this function only when the acquisition mode is Normal, Hi Res, or Average. If the acquisition type is not one of the listed types, the function returns the Invalid Acquisition Type error.

You call the Fetch Waveform function to obtain the waveforms for each of the remaining enabled channels without initiating another acquisition. After this function executes, each element in the `WaveformArray` parameter is either a voltage or a value indicating that the oscilloscope could not sample a voltage.

The C end-user configures the interpolation method the oscilloscope uses with the `IviScope_ConfigureInterpolation` function. The COM end-user uses the `Acquisition.Interpolation` property. If interpolation is disabled, the oscilloscope does not interpolate points in the waveform. If the oscilloscope cannot sample a value for a point in the waveform, the driver sets the corresponding element in the `WaveformArray` to an IEEE-defined NaN (Not a Number) value and the function returns Invalid Wfm Element.

Use the Is Waveform Element Invalid function to test each element in the Waveform Array parameter for an invalid waveform element.

COM Method Prototype

```
HRESULT Measurements.Item().ReadWaveform ([in] LONG MaxTimeMilliseconds,
                                  [in,out] SAFEARRAY(DOUBLE) *WaveformArray,
                                  [in,out] DOUBLE *InitialX,
                                  [in,out] DOUBLE *XIncrement);
```

C Prototype

```
ViStatus IviScope_ReadWaveform (ViSession vi,
                                ViConstString Channel,
                                ViInt32 WaveformSize,
                                ViInt32 MaxTimeMilliseconds,
                                ViReal64 WaveformArray[],
                                ViInt32 *ActualPoints,
                                ViReal64 *InitialX,
                                ViReal64 *XIncrement);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Channel	Name of the channel from which to read a waveform.	ViConstString
WaveformSize	Specifies the number of elements in the waveform array.	ViInt32
MaxTime Milliseconds	Specifies the maximum time the end-user allows for this function to complete in milliseconds.	ViInt32

Outputs	Description	Base Type
WaveformArray	A user-allocated (for IVI-C) or driver-allocated (for IVI-COM) buffer into which the driver stores the waveform it reads. The units for the individual array elements are volts.	ViReal64[]
ActualPoints	Contains the number of points the driver actually places in the waveform array.	ViInt32
InitialX	Contains the time of the first point in the waveform. The value is with respect to the trigger and is in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger.	ViReal64
XIncrement	Contains the effective time between points in the waveform. The units are seconds.	ViReal64

Defined Values for the MaxTimeMilliseconds Parameter

Name	*Description*		
		Language	*Identifier*
Max Time Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		C	IVISCOPE_VAL_MAX_TIME_IMMEDIATE
		COM	IviScopeTimeOutImmediate
Max Time Infinite	The function waits indefinitely for the measurement to complete.		
		C	IVISCOPE_VAL_MAX_TIME_INFINITE
		COM	IviScopeTimeOutInfinite

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 Waveform Element	Warning: One of the elements in the waveform array is invalid.
Invalid Acquisition Type	Error: Invalid acquisition type
Channel Not Enabled	Error: Specified channel is not enabled for acquisition.
Max Time Exceeded	Error: Maximum time exceeded before the operation completed.

Compliance Notes

The specific instrument driver is not required to support any of the defined values for the `MaxTimeMilliseconds` parameter.

4.3.16 Sample Rate (IVI-C only)

Description

This function returns the effective sample rate of the acquired waveform using the current configuration in samples per second.

COM Method Prototype

N/A

(use the `Acquisition.SampleRate` property)

C Prototype

```
ViStatus IviScope_SampleRate (ViSession Vi,
                              ViReal64 *SampleRate);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Outputs	Description	Base Type
SampleRate	Returns the effective sample rate of the acquired waveform the oscilloscope acquires for each channel. The driver returns the value held in the Horizontal Sample Rate attribute. See the Horizontal Sample Rate attribute for a complete description.	ViReal64

Return Values

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

4.4 IviScope Behavior Model

The following behavior diagram shows relationships between IviScopeBase capabilities and oscilloscope behavior.

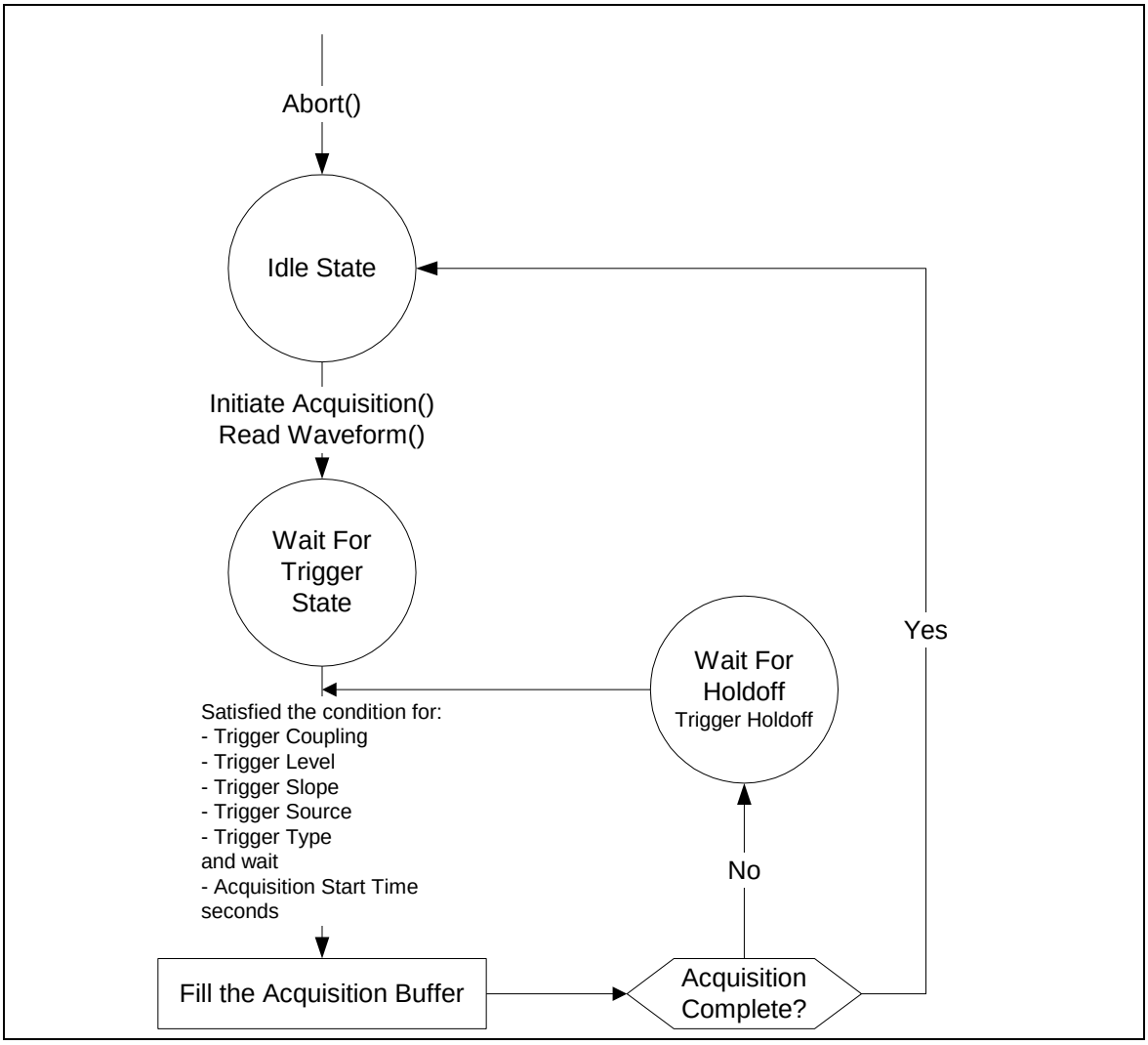


Figure 4-5. IviScope Behavior Model

Typically the user configures the oscilloscope while it is in the *Idle* state. You can configure the oscilloscope by calling the high-level configure channel, configure acquisition type, configure acquisition record, configure trigger, configure trigger coupling, and configure edge trigger source functions.

To acquire waveforms, the IviScope class presents the high-level read waveform function, as well as the low-level functions initiate acquisition, acquisition status, fetch waveform, and abort.

The Read Waveform function initiates a waveform acquisition and returns the acquired waveform after the oscilloscope has returned to the *Idle* state.

The Initiate Acquisition, Fetch Waveform, and Abort functions give the user lower-level control over the measurement process. Initiate Acquisition initiates a waveform acquisition and moves the instrument into the *Wait-For-Trigger* state. The type of trigger is configured with the Trigger sub-system attributes or with the configure edge trigger source function.

If the acquisition start time is negative, the first point in the waveform record occurs prior to the trigger event. When the trigger event occurs, the waveform record contains the amount of pre-trigger data that corresponds to the acquisition start time. The scope leaves the *Wait-for-Trigger* state and acquires the remaining points in the waveform record.

If the acquisition start time equals zero, the first point in the waveform record occurs at the time of the trigger event. When the trigger event occurs, the scope leaves the *Wait-for-Trigger* state and acquires all the points in the waveform record.

If the acquisition start time is greater than zero, the first point in the waveform record occurs after the trigger event. When the trigger event occurs, the scope leaves the wait-for-trigger state, waits a length of time that is equal to the acquisition start time, and acquires all the points in the waveform record.

If the oscilloscope was able to fill all of the points in the waveform in real-time it then returns to the Idle state. However, if the oscilloscope must acquire multiple waveforms in equivalent-time sampling to build up the waveform record, it then moves to the Wait-For-Holdoff state. The oscilloscope then waits until the hold-off time expires before moving to the Wait-For-Trigger.

Note that the hold-off time is measured from the moment the oscilloscope exits the *Wait-for-Trigger* state, not from the moment when the oscilloscope enters the *Wait-for-Holdoff* state.

After the instrument meets its acquisition complete criterion, the oscilloscope returns to the *Idle* state. (This criterion is typically 95-98% of the acquisition record; there may be instrument specific attributes that allow you to configure the completion criterion.) You can use the Acquisition Status function to determine if the acquisition is complete or is still in progress.

The Fetch Waveform function is used to return a waveform from a previously initiated measurement. The Read Waveform and Fetch Waveform functions return the following parameters:

- a waveform array
- the time of the first point in the waveform array in relationship to the trigger event
- the effective time interval between points in the array.

5. IviScopeInterpolation Extension Group

5.1 IviScopeInterpolation Overview

The IviScopeInterpolation extension group defines extensions for oscilloscopes capable of interpolating values in the waveform record that the oscilloscope's acquisition sub-system was unable to digitize.

5.2 IviScopeInterpolation Attributes

The IviScopeInterpolation capability group defines the following attribute:

- Interpolation

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

5.2.1 Interpolation

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Interpolation (IVI-C only)

COM Property Name

Acquisition.Interpolation

COM Enumeration Name

IviScopeInterpolationEnum

C Constant Name

IVISCOPE_ATTR_INTERPOLATION

Description

Specifies the interpolation method the oscilloscope uses when it cannot resolve a voltage for every point in the waveform record.

Defined Values

Name	*Description*		
		Language	*Identifier*
No Interpolation	The oscilloscope does not interpolate points in the waveform. Instead, the driver sets every element in the waveform record for which the oscilloscope cannot receive a value to an IEEE-defined NaN (Not-a-Number) value. Use the Is Waveform Element Invalid function to determine if the waveform record element is invalid.		
		C	IVISCOPE_VAL_NO_INTERPOLATION
		COM	IviScopeInterpolationNone
Sine X	The oscilloscope uses a sin(x)/x calculation to interpolate a value when it cannot resolve a voltage in the waveform record.		
		C	IVISCOPE_VAL_SINE_X
		COM	IviScopeInterpolationSineX
Linear	The oscilloscope uses a linear approximation to interpolate a value when it cannot resolve a voltage in the waveform record.		
		C	IVISCOPE_VAL_LINEAR
		COM	IviScopeInterpolationLinear

Compliance Notes

1. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_INTERPOLATION_CLASS_EXT_BASE and less than IVISCOPE_VAL_INTERPOLATION_SPECIFIC_EXT_BASE.
2. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_INTERPOLATION_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 Interpolation Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of Interpolation Specific Ext Base, IVISCOPE_VAL_INTERPOLATION_SPECIFIC_EXT_BASE and IVISCOPE_VAL_INTERPOLATION_CLASS_EXT_BASE.

5.3 IviScopeInterpolation Functions

The IviScopeInterpolation capability group defines the following function for IVI-C driver only:

- Configure Interpolation (IVI-C only)

This section describes the behavior and requirements of this function.

5.3.1 Configure Interpolation (IVI-C only)

Description

This function configures the interpolation method the oscilloscope uses when it cannot sample a voltage for a point in the waveform record.

COM Method Prototype

`N/A`
(use the `Acquisition.Interpolation` property)

C Prototype

```
ViStatus IviScope_ConfigureInterpolation (ViSession Vi,
                                          ViInt32 Interpolation);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Interpolation`	Specifies interpolation method the oscilloscope uses when it cannot sample a voltage for a point in the waveform record. The driver sets the Interpolation attribute to this value. See the Interpolation attribute for a complete description and defined values.	`ViInt32`

Return Values

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

5.4 IviScopeInterpolation Behavior Model

The IviScopeInterpolation group uses the behavior model defined by the IviScopeBase Capabilities.

6. IviScopeTVTrigger Extension Group

6.1 IviScopeTVTrigger Overview

The IviScopeTVTrigger extension group defines extensions for oscilloscopes capable of triggering on TV signals.

6.2 IviScopeTVTrigger Attributes

The IviScopeInterpolation capability group defines the following attributes:

- TVTrigger Event
- TVTrigger Line Number
- TVTrigger Polarity
- TVTrigger Signal Format

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

6.2.1 TV Trigger Event

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

COM Property Name

Trigger.TV.Event

COM Enumeration Name

IviScopeTVTriggerEventEnum

C Constant Name

IVISCOPE_ATTR_TV_TRIGGER_EVENT

Description

Specifies the event on which the oscilloscope triggers.

Defined Values

Name	*Description*		
		Language	*Identifier*
TV Event Field 1	Sets the oscilloscope to trigger on field 1 of the video signal.		
		C	IVISCOPE_VAL_TV_EVENT_FIELD1
		COM	IviScopeTVTriggerEventField1
TV Event Field 2	Sets the oscilloscope to trigger on field 2 of the video signal.		
		C	IVISCOPE_VAL_TV_EVENT_FIELD2
		COM	IviScopeTVTriggerEventField2
TV Event Any Field	Sets the oscilloscope to trigger on any field.		
		C	IVISCOPE_VAL_TV_EVENT_ANY_FIELD
		COM	IviScopeTVTriggerEventAnyField
TV Event Any Line	Sets the oscilloscope to trigger on any line.		
		C	IVISCOPE_VAL_TV_EVENT_ANY_LINE
		COM	IviScopeTVTriggerEventAnyLine
TV Event Line Number	Sets the oscilloscope to trigger on a specific line number you specify with the TV Trigger Line Number attribute.		
		C	IVISCOPE_VAL_TV_EVENT_LINE_NUMBER
		COM	IviScopeTVEventLineNumber

Compliance Notes

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

2. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TV_TRIGGER_EVENT_SPECIFIC_EXT_BASE.
3. If an IVI Class-Compliant specific driver implements the TV Event Line Number value, it shall also implement the TV Trigger Line Number attribute. An IVI-C specific driver shall also implement the IviScope_ConfigureTVTriggerLineNumber function.
4. 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 TV Trigger Event Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of TV Trigger Event Specific Ext Base, IVISCOPE_VAL_TV_TRIGGER_EVENT_SPECIFIC_EXT_BASE and IVISCOPE_VAL_TV_TRIGGER_EVENT_CLASS_EXT_BASE.

6.2.2 TV Trigger Line Number

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

COM Property Name

Trigger.TV.LineNumber

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_TV_TRIGGER_LINE_NUMBER

Description

Specifies the line on which the oscilloscope triggers. The driver uses this attribute when the TV Trigger Event is set to TV Event Line Number. The line number setting is independent of the field. This means that to trigger on the first line of the second field, the user must configure the line number to the value of 263 (if we presume that field one had 262 lines).

Compliance Notes

1. An IVI Class-Compliant specific driver shall implement this attribute only if it implements the TV Event Line Number value for the TV Trigger Event attribute.
2. If an IVI-C specific driver implements this attribute, then it shall also implement the IviScope_ConfigureTVTriggerLineNumber function.

6.2.3 TV Trigger Polarity

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

COM Property Name

Trigger.TV.Polarity

COM Enumeration Name

IviScopeTVTriggerPolarityEnum

C Constant Name

IVISCOPE_ATTR_TV_TRIGGER_POLARITY

Description

Specifies the polarity of the TV signal.

Defined Values

Name	*Description*		
		Language	*Identifier*
TV Positive	Configures the oscilloscope to trigger on a positive video sync pulse.		
		C	IVISCOPE_VAL_TV_POSITIVE
		COM	IviScopeTVTriggerPolarityPositive
TV Negative	Configures the oscilloscope to trigger on a negative video sync pulse.		
		C	IVISCOPE_VAL_TV_NEGATIVE
		COM	IviScopeTVTriggerPolarityNegative

6.2.4 TV Trigger Signal Format

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

COM Property Name

Trigger.TV.SignalFormat

COM Enumeration Name

IviScopeTVSignalFormatEnum

C Constant Name

IVISCOPE_ATTR_TV_TRIGGER_SIGNAL_FORMAT

Description

Specifies the format of TV signal on which the oscilloscope triggers

Defined Values

Name	*Description*		
		Language	*Identifier*
NTSC	Configures the oscilloscope to trigger on the NTSC signal format.		
		C	IVISCOPE_VAL_NTSC
		COM	IviScopeTVSignalFormatNTSC
PAL	Configures the oscilloscope to trigger on the PAL signal format		
		C	IVISCOPE_VAL_PAL
		COM	IviScopeTVSignalFormatPAL
SECAM	Configures the oscilloscope to trigger on the SECAM signal format		
		C	IVISCOPE_VAL_SECAM
		COM	IviScopeTVSignalFormatSECAM

Compliance Notes

1. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_SIGNAL_FMT_CLASS_EXT_BASE and less than IVISCOPE_VAL_SIGNAL_FMT_SPECIFIC_EXT_BASE.
2. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TV_TRIGGER_SIGNAL_FMT_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 TV Trigger Signal Format Specific Ext Base.

See Section 20, *IviScope Attribute Value Definitions*, for the definitions of TV Trigger Signal Format Specific Ext Base, IVISCOPE_VAL_TV_TRIGGER_SIGNAL_FMT_SPECIFIC_EXT_BASE and IVISCOPE_VAL_TV_TRIGGER_SIGNAL_FMT_CLASS_EXT_BASE.

6.3 IviScopeTVTrigger Functions

The IviScopeTVTrigger capability group defines the following functions:

- Configure TV Trigger Line Number (IVI-C only)
- Configure TV Trigger Source

This section describes the behavior and requirements of each function.

6.3.1 Configure TV Trigger Line Number (IVI-C only)

Description

This function configures the TV line upon which the oscilloscope triggers. The line number is absolute and not relative to the field of the TV signal.

This function affects instrument behavior only if the trigger type is set to the TV Trigger value and the TV trigger event is set to the TV Event Line Number value. Call the Configure TV Trigger Source function to set the TV trigger event before calling this function.

COM Method Prototype

`N/A`
(use the `Trigger.TV.LineNumber` property)

C Prototype

```
ViStatus IviScope_ConfigureTVTriggerLineNumber (ViSession Vi,
                                                ViInt32 TVLineNumber);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`TVLineNumber`	Specifies the TV trigger line number. The driver sets the TV Trigger Line Number attribute to this value. See the TV Trigger Line Number attribute description for details.	`ViInt32`

Return Values

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

Compliance Notes

If an IVI-C specific driver implements this function, then this driver shall implement the TV Trigger Line Number attribute and also it shall implement the TV Event Line Number value for the TV Trigger Event attribute.

6.3.2 Configure TV Trigger Source

Description

This function configures the oscilloscope for TV triggering. It configures the TV signal format, the event and the signal polarity.

This function affects instrument behavior only if the trigger type is TV Trigger. Set the Trigger Type and Trigger Coupling before calling this function.

COM Method Prototype

```
HRESULT Trigger.TV.Configure ([in] BSTR Source,
                              [in] IviScopeTVSignalFormatEnum SignalFormat,
                              [in] IviScopeTVTriggerEventEnum Event,
                              [in] IviScopeTVTriggerPolarityEnum Polarity);
```

C Prototype

```
ViStatus IviScope_ConfigureTVTriggerSource (ViSession Vi,
                                            ViConstString Source,
                                            ViInt32 TVSignalFormat,
                                            ViInt32 TVEvent,
                                            ViInt32 TVPolarity);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Source	Specifies the trigger source. The driver uses this value to se the Trigger Source attribute. See the Trigger Source attribute description for more information.	ViConstString
TVSignalFormat	Specifies the TV trigger signal format. The driver uses this value to set the TV Trigger Signal Format attribute. See the TV Trigger Signal Format attribute description for more information.	ViInt32
TVEvent	Specifies the TV trigger event. The driver uses this value to set the TV Trigger Event attribute. See the TV Trigger Event attribute description for more information.	ViInt32
TVPolarity	Specifies the polarity of the TV trigger. The driver uses this value to set the TV Trigger Polarity attribute. See the TV Trigger Polarity attribute description for more information.	ViInt32

Return Values

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

6.4 IviScopeTVTrigger Behavior Model

The IviScopeTVTrigger group uses the behavior model defined by the IviScopeBase Capabilities.

6.5 IviScopeTVTrigger Compliance Notes

IVI Class-Compliant specific drivers that implement this extension group shall implement the TVTrigger value for the Trigger Type attribute in the IviScopeBase capabilities group.

7. IviScopeRuntTrigger Extension Group

7.1 IviScopeRuntTrigger Overview

In addition to the fundamental capabilities, the IviScopeRuntTrigger extension group defines extensions for oscilloscopes with the capability to trigger on “runt” pulses.

A runt condition occurs when the oscilloscope detects a positive or negative going pulse that crosses one voltage threshold but fails to cross a second threshold before re-crossing the first. The figure below shows both positive and negative runt polarities.

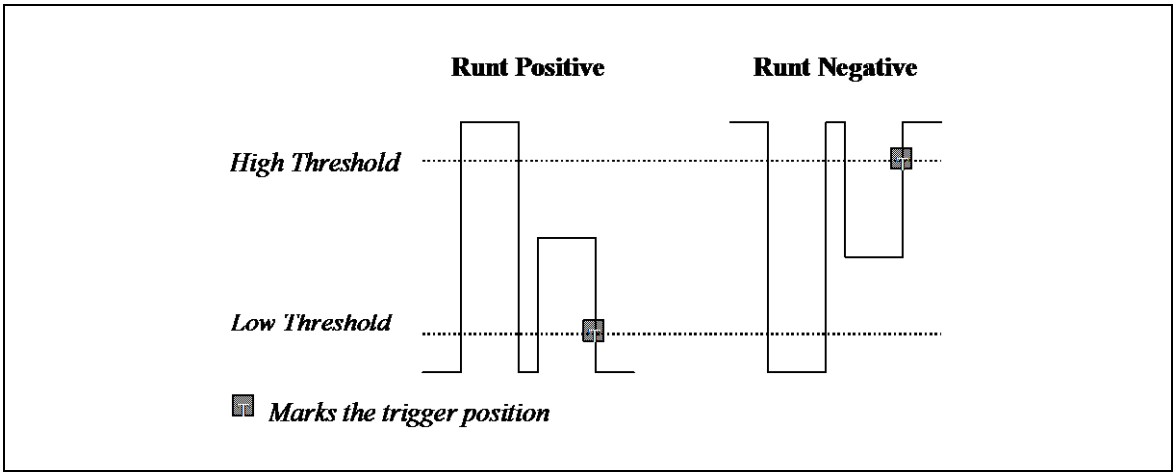


Figure 7-1. Runt Triggers

With the IviScopeRuntTrigger extension group the end-user can select whether a positive runt, negative runt, or either triggers the acquisition.

7.2 IviScopeRuntTrigger Attributes

The IviScopeRuntTrigger capability group defines the following attributes:

- Runt High Threshold
- Runt Low Threshold
- Runt Polarity

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

7.2.1 Runt High Threshold

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

COM Property Name

Trigger.Runt.ThresholdHigh

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_RUNT_HIGH_THRESHOLD

Description

Specifies the high threshold the oscilloscope uses for runt triggering. The units are volts.

7.2.2 Runt Low Threshold

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

COM Property Name

Trigger.Runt.ThresholdLow

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_RUNT_LOW_THRESHOLD

Description

Specifies the low threshold the oscilloscope uses for runt triggering. The units are volts.

7.2.3 Runt Polarity

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

COM Property Name

Trigger.Runt.Polarity

COM Enumeration Name

IviScopeRuntPolarityEnum

C Constant Name

IVISCOPE_ATTR_RUNT_POLARITY

Description

Specifies the polarity of the runt that triggers the oscilloscope.

Defined Values

Name	*Description*		
		Language	*Identifier*
Runt Positive	The oscilloscope triggers on a positive runt. A positive runt occurs when a rising edge crosses the low runt threshold and does not cross the high runt threshold before re-crossing the low runt threshold.		
		C	IVISCOPE_VAL_RUNT_POSITIVE
		COM	IviScopeRuntPolarityPositive
Runt Negative	The oscilloscope triggers on a negative runt. A negative runt occurs when a falling edge crosses the high runt threshold and does not cross the low runt threshold before re-crossing the high runt threshold.		
		C	IVISCOPE_VAL_RUNT_NEGATIVE
		COM	IviScopeRuntPolarityNegative
Runt Either	The oscilloscope triggers on either a positive or negative runt.		
		C	IVISCOPE_VAL_RUNT_EITHER
		COM	IviScopeRuntPolarityEither

7.3 IviScopeRuntTrigger Functions

The IviScopeRuntTrigger capability group defines the following function:

- Configure Runt Trigger Source

This section describes the behavior and requirements of this function.

7.3.1 ConfigureRuntTriggerSource

Description

This function configures the runt trigger. A runt trigger occurs when the trigger signal crosses one of the runt thresholds twice without crossing the other runt threshold. The end-user specifies the runt thresholds with the `RuntLowThreshold` and `RuntHighThreshold` parameters. The end-user specifies the polarity of the runt with the `RuntPolarity` parameter.

This function affects instrument behavior only if the trigger type is Runt Trigger. Set the trigger type and trigger coupling before calling this function.

COM Method Prototype

```
HRESULT Trigger.Runt.Configure ([in] BSTR Source,
                                [in] DOUBLE ThresholdLow,
                                [in] DOUBLE ThresholdHigh,
                                [in] IviScopeRuntPolarityEnum Polarity);
```

C Prototype

```
ViStatus IviScope_ConfigureRuntTriggerSource (ViSession Vi,
                                              ViConstString Source,
                                              ViReal64 RuntLowThreshold,
                                              ViReal64 RuntHighThreshold,
                                              ViInt32 RuntPolarity);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Source`	Specifies the trigger source. The driver uses this value to set the Trigger Source attribute. See the attribute description for more information.	`ViConstString`
`RuntLowThreshold`	Sets the runt triggering low threshold in volts. See attribute Runt Low Threshold for a complete description.	`ViReal64`
`RuntHighThreshold`	Sets the runt triggering high threshold in volts. See attribute Runt High Threshold for a complete description.	`ViReal64`
`RuntPolarity`	Sets the runt polarity. See attribute Runt Polarity for a complete description and defined values.	`ViInt32`

Return Values

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

7.4 IviScopeRuntTrigger Behavior Model

The IviScopeRuntTrigger group uses the behavior model defined by the IviScopeBase Capabilities.

7.5 IviScopeRuntTrigger Compliance Notes

IVI Class-Compliant specific drivers that implement this extension group shall implement the Runt Trigger value for the Trigger Type attribute in the IviScopeBase capabilities group.

8. IviScopeGlitchTrigger Extension Group

8.1 IviScopeGlitchTrigger Overview

In addition to the fundamental capabilities, the IviScopeGlitchTrigger extension group defines extensions for oscilloscopes that can trigger on a "glitch" pulses.

A glitch occurs when the oscilloscope detects a pulse width that is less than or a greater than a specified glitch duration. The figure below shows both positive and negative glitches for the "less than" condition as well as the positive "greater than" glitch.

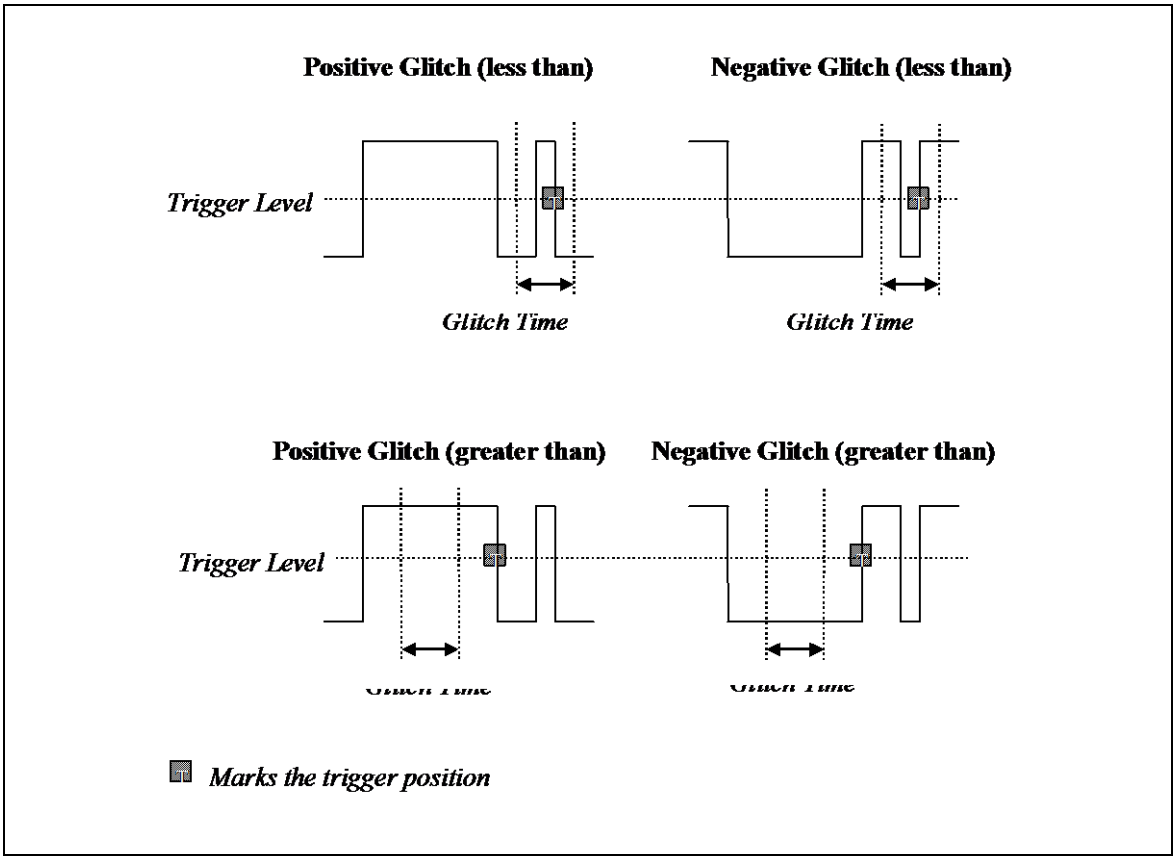


Figure 8-1. Glitch Triggers

With the IviScopeGlitchTrigger extension group the end-user can select whether a positive glitch, negative glitch, or either triggers the acquisition.

8.2 IviScopeGlitchTrigger Attributes

The IviScopeGlitchTrigger capability group defines the following attributes:

- Glitch Condition
- Glitch Polarity
- Glitch Width

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

8.2.1 Glitch Condition

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

COM Property Name

Trigger.Glitch.Condition

COM Enumeration Name

IviScopeGlitchConditionEnum

C Constant Name

IVISCOPE_ATTR_GLITCH_CONDITION

Description

Specifies the glitch condition. This attribute determines whether the glitch trigger happens when the oscilloscope detects a pulse with a width less than or greater than the width value

Defined Values

Name	*Description*		
		Language	*Identifier*
Glitch Greater Than	The oscilloscope triggers when the pulse width is greater than the value you specify with the Glitch Width attribute.		
		C	IVISCOPE_VAL_GLITCH_GREATER_THAN
		COM	IviScopeGlitchConditionGreaterThan
Glitch Less Than	The oscilloscope triggers when the pulse width is less than the value you specify with the Glitch Width attribute.		
		C	IVISCOPE_VAL_GLITCH_LESS_THAN
		COM	IviScopeGlitchConditionLessThan

8.2.2 Glitch Polarity

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

COM Property Name

Trigger.Glitch.Polarity

COM Enumeration Name

IviScopeGlitchPolarityEnum

C Constant Name

IVISCOPE_ATTR_GLITCH_POLARITY

Description

Specifies the polarity of the glitch that triggers oscilloscope.

Defined Values

Name	*Description*		
		Language	*Identifier*
Glitch Positive	The oscilloscope triggers on a positive glitch.		
		C	IVISCOPE_VAL_GLITCH_POSITIVE
		COM	IviScopeGlitchPolarityPositive
Glitch Negative	The oscilloscope triggers on a negative glitch.		
		C	IVISCOPE_VAL_GLITCH_NEGATIVE
		COM	IviScopeGlitchPolarityNegative
Glitch Either	The oscilloscope triggers on either a positive or negative glitch.		
		C	IVISCOPE_VAL_GLITCH_EITHER
		COM	IviScopeGlitchPolarityEither

8.2.3 Glitch Width

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

COM Property Name

Trigger.Glitch.Width

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_GLITCH_WIDTH

Description

Specifies the glitch width. The units are seconds. The oscilloscope triggers when it detects a pulse with a width less than or greater than this value, depending on the Glitch Condition attribute.

8.3 IviScopeGlitchTrigger Functions

The IviScopeGlitchTrigger capability group defines the following function:

- Configure Glitch Trigger Source

This section describes the behavior and requirements of this function.

8.3.1 Configure Glitch Trigger Source

Description

This function configures the glitch trigger. A glitch trigger occurs when the trigger signal has a pulse with a width that is less than or greater than the glitch width. The end user specifies which comparison criterion to use with the `GlitchCondition` parameter. The end-user specifies the glitch width with the `GlitchWidth` parameter. The end-user specifies the polarity of the pulse with the `GlitchPolarity` parameter. The trigger does not actually occur until the edge of a pulse that corresponds to the `GlitchWidth` and `GlitchPolarity` crosses the threshold the end-user specifies in the `TriggerLevel` parameter.

This function affects instrument behavior only if the trigger type is Glitch Trigger. Set the trigger type and trigger coupling before calling this function.

COM Method Prototype

```
HRESULT Trigger.Glitch.Configure ([in] BSTR Source,
                                  [in] DOUBLE Level,
                                  [in] DOUBLE Width,
                                  [in] IviScopeGlitchPolarityEnum Polarity,
                                  [in] IviScopeGlitchConditionEnum Condition)
```

C Prototype

```
ViStatus IviScope_ConfigureGlitchTriggerSource (ViSession Vi,
                                                ViConstString Source,
                                                ViReal64 Level,
                                                ViReal64 GlitchWidth,
                                                ViInt32 GlitchPolarity,
                                                ViInt32 GlitchCondition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Source`	Specifies the trigger source. The driver uses this value to se the Trigger Source attribute. See the attribute description fo more information.	`ViConstString`
`Level`	Specifies the trigger level. The driver uses this value to set the Trigger Level attribute. See the attribute description for more information.	`ViReal64`
`GlitchWidth`	Specifies the glitch triggering glitch width in seconds. The driver uses this value to set the Glitch Width attribute. See the attribute description for more information	`ViReal64`
`GlitchPolarity`	Specifies the glitch polarity. The driver uses this value to se the Glitch Polarity attribute. See the attribute description for more information.	`ViInt32`
`GlitchCondition`	Specifies the glitch condition. The driver uses this value to set the Glitch Condition attribute. See the attribute description for more information.	`ViInt32`

Return Values

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

8.4 IviScopeGlitchTrigger Behavior Model

The IviScopeGlitchTrigger group uses the behavior model defined by the IviScopeBase Capabilities.

8.5 IviScopeGlitchTrigger Compliance Notes

IVI Class-Compliant specific drivers that implement this extension group shall implement the Glitch Trigger value for the Trigger Type attribute in the IviScopeBase capabilities group.

9. IviScopeWidthTrigger Extension Group

9.1 IviScopeWidthTrigger Overview

In addition to the fundamental capabilities, the IviScopeWidthTrigger extension group defines extensions for oscilloscopes capable of triggering on user-specified pulse widths.

Width triggering occurs when the oscilloscope detects a positive or negative pulse with a width between, or optionally outside, the user-specified thresholds. The figure below shows positive and negative pulses that fall within the user-specified thresholds.

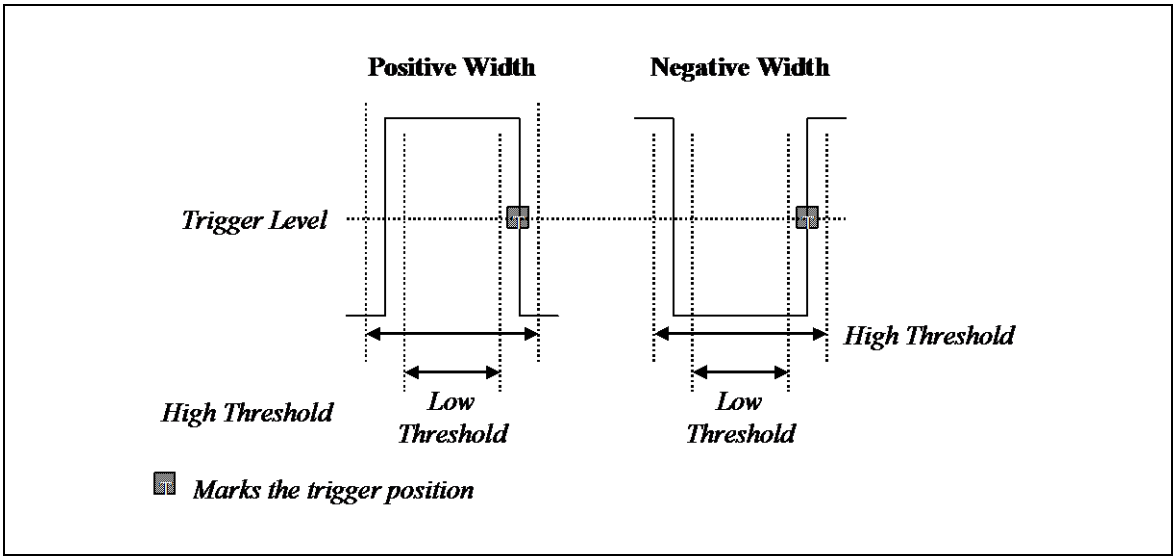


Figure 9-1. Width Triggers Within the Thresholds

The figure below shows positive and negative pulses that are not inside the user-specified thresholds.

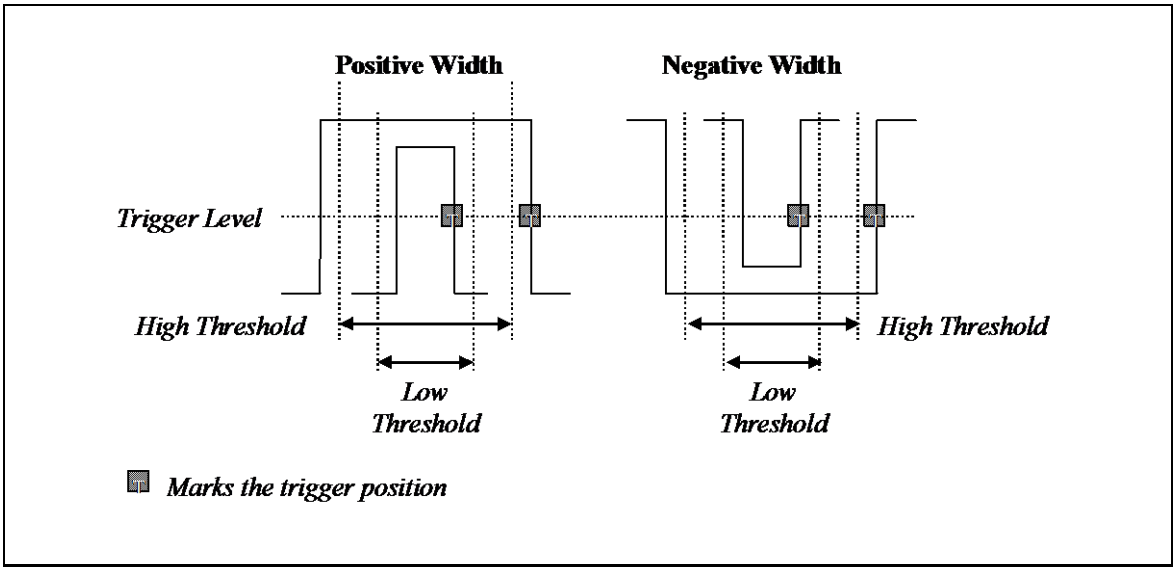


Figure 9-2. Width Triggers Outside the Thresholds

9.2 IviScopeWidthTrigger Attributes

The IviScopeWidthTrigger capability group defines the following attributes:

- Width Condition
- Width High Threshold
- Width Low Threshold
- Width Polarity

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

9.2.1 Width Condition

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

COM Property Name

Trigger.Width.Condition

COM Enumeration Name

IviScopeWidthConditionEnum

C Constant Name

IVISCOPE_ATTR_WIDTH_CONDITION

Description

Specifies whether a pulse that is within or outside the high and low thresholds triggers the oscilloscope. The end-user specifies the high and low thresholds with the Width High Threshold and Width Low Threshold attributes.

Defined Values

Name	*Description*		
		Language	*Identifier*
Width Within	Configures the oscilloscope to trigger on pulses that have a width that is less than the high threshold and greater than the low threshold. The end-user specifies the high and low thresholds with the Width High Threshold and Width Low Threshold attributes.		
		C	IVISCOPE_VAL_WIDTH_WITHIN
		COM	IviScopeWidthConditionWithin
Width Outside	Configures the oscilloscope to trigger on pulses that have a width that is either greater than the high threshold or less than a low threshold. The end-user specifies the high and low thresholds with the Width High Threshold and Width Low Threshold attributes.		
		C	IVISCOPE_VAL_WIDTH_OUTSIDE
		COM	IviScopeWidthConditionOutside

9.2.2 Width High Threshold

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

COM Property Name

Trigger.Width.ThresholdHigh

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_WIDTH_HIGH_THRESHOLD

Description

Specifies the high width threshold time in seconds.

9.2.3 Width Low Threshold

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

COM Property Name

Trigger.Width.ThresholdLow

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_WIDTH_LOW_THRESHOLD

Description

Specifies the low width threshold time in seconds.

9.2.4 Width Polarity

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

COM Property Name

Trigger.Width.Polarity

COM Enumeration Name

IviScopeWidthPolarityEnum

C Constant Name

IVISCOPE_ATTR_WIDTH_POLARITY

Description

Specifies the polarity of the pulse that triggers the oscilloscope.

Defined Values

Name	***Description***		
		Language	***Identifier***
Width Positive	Configures the oscilloscope to trigger on positive pulses that have a width that meets the condition the user specifies with the Width Condition attribute.		
		C	IVISCOPE_VAL_WIDTH_POSITIVE
		COM	IviScopeWidthPolarityPositive
Width Negative	Configures the oscilloscope to trigger on negative pulses that have a width that meets the condition the user specifies with the Width Condition attribute.		
		C	IVISCOPE_VAL_WIDTH_NEGATIVE
		COM	IviScopeWidthPolarityNegative
Width Either	Configures the oscilloscope to trigger on either positive or negative pulses that have a width that meets the condition the user specifies with the Width Condition attribute.		
		C	IVISCOPE_VAL_ WIDTH_EITHER
		COM	IviScopeWidthPolarityEither

9.3 IviScopeWidthTrigger Functions

The IviScopeWidthTrigger capability group defines the following function:

- Configure Width Trigger Source

This section describes the behavior and requirements of this function.

9.3.1 Configure Width Trigger Source

Description

This function configures the width trigger. A width trigger occurs when the oscilloscope detects a positive or negative pulse with a width between, or optionally outside, the width thresholds. The end-user specifies the width thresholds with the `WidthLowThreshold` and `WidthHighThreshold` parameters. The end-user specifies whether the oscilloscope triggers on pulse widths that are within or outside the width thresholds with the `WidthCondition` parameter. The end-user specifies the polarity of the pulse with the `WidthPolarity` parameter. The trigger does not actually occur until the edge of a pulse that corresponds to the `WidthLowThreshold`, `WidthHighThreshold`, `WidthCondition`, and `WidthPolarity` crosses the threshold the end-user specifies with the `TriggerLevel` parameter.

This function affects instrument behavior only if the trigger type is Width Trigger. Set the trigger type and trigger coupling before calling this function.

COM Method Prototype

```
HRESULT Trigger.Width.Configure ([in] BSTR Source,
                                 [in] DOUBLE Level,
                                 [in] DOUBLE ThresholdLow,
                                 [in] DOUBLE ThresholdHigh,
                                 [in] IviScopeWidthPolarityEnum Polarity,
                                 [in] IviScopeWidthConditionEnum Condition);
```

C Prototype

```
ViStatus IviScope_ConfigureWidthTriggerSource (ViSession Vi,
                                               ViConstString Source,
                                               ViReal64 Level,
                                               ViReal64 WidthLowThreshold,
                                               ViReal64 WidthHighTreshold,
                                               ViInt32 WidthPolarity,
                                               ViInt32 WidthCondition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Source`	Specifies the trigger source. The driver uses this value to se the Trigger Source attribute. See the attribute description fo more information.	`ViConstString`
`Level`	Trigger Level. The driver uses this value to set the Trigger Level attribute. See the attribute description for more information.	`ViReal64`
`WidthLowThreshold`	Sets the width triggering low threshold in seconds. The driver uses this value to set the Width Low Threshold attribute. See the attribute description for more information.	`ViReal64`
`WidthHighTreshold`	Sets the width triggering high threshold in seconds. The driver uses this value to set the Width High Threshold attribute. See the attribute description for more information.	`ViReal64`
`WidthPolarity`	Sets the width polarity. The driver uses this value to set the Width Polarity attribute. See the attribute description for more information.	`ViInt32`

WidthCondition	Specifies whether a pulse that is within or outside the user-specified thresholds trigger waveform acquisition. The driver uses this value to set the Width Condition attribute. See the attribute description for more information.	ViInt32

Return Values

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

9.4 IviScopeWidthTrigger Behavior Model

The IviScopeWidthTrigger group uses the behavior model defined by the IviScopeBase Capabilities.

9.5 IviScopeWidthTrigger Compliance Notes

IVI Class-Compliant specific drivers that implement this extension group shall implement the Width Trigger value for the Trigger Type attribute in the IviScopeBase capabilities group.

10. IviScopeAcLineTrigger Extension Group

10.1 IviScopeAcLineTrigger Overview

In addition to the fundamental capabilities, the IviScopeAcLineTrigger extension group defines extensions for oscilloscopes that are capable of synchronizing the trigger with the AC Line.

AC Line triggering occurs when the oscilloscope detects a positive zero crossing, negative zero crossing, or optionally either positive or negative zero crossing on the network supply voltage.

10.2 IviScopeAcLineTrigger Attributes

The IviScopeAcLineTrigger capability group defines the following attributes:

- AC Line Trigger Slope

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

10.2.1 AC Line Trigger Slope

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure AC Line Trigger Slope (IVI-C only)

COM Property Name

Trigger.ACLine.Slope

COM Enumeration Name

IviScopeACLineSlopeEnum

C Constant Name

IVISCOPE_ATTR_AC_LINE_TRIGGER_SLOPE

Description

Specifies the slope of the zero crossing upon which the scope triggers.

Defined Values

Name	*Description*		
		Language	*Identifier*
AC Line Positive	Configures the oscilloscope to trigger on positive slope zero crossings of the network supply voltage.		
		C	IVISCOPE_VAL_AC_LINE_POSITIVE
		COM	IviScopeACLinePositive
AC Line Negative	Configures the oscilloscope to trigger on negative slope zero crossings of the network supply voltage.		
		C	IVISCOPE_VAL_AC_LINE_NEGATIVE
		COM	IviScopeACLineNegative
AC Line Either	Configures the oscilloscope to trigger on either positive or negative slope zero crossings of the network supply voltage.		
		C	IVISCOPE_VAL_AC_LINE_EITHER
		COM	IviScopeACLineEither

Compliance Notes

The driver shall implement the AC Line Either value for this attribute.

10.3 IviScopeAcLineTrigger Functions

The IviScopeAcLineTrigger capability group defines the following function:

- Configure AC Line Trigger Slope (IVI-C only)

This section describes the behavior and requirements of this function.

10.3.1 Configure AC Line Trigger Slope (IVI-C only)

Description

This function configures the slope of the AC Line trigger.

This function affects instrument behavior only if the trigger type is AC Line Trigger. Call the Configure Trigger function to set the trigger type before calling this function.

COM Method Prototype

`N/A`
(use the `Trigger.ACLine.Slope` property)

C Prototype

```
ViStatus IviScope_ConfigureAcLineTriggerSlope (ViSession Vi,
                                               ViInt32 ACLineSlope);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`ACLineSlope`	Specifies whether an oscilloscope triggers on a zero crossing with a positive, negative, or either slope of the network supply voltage. The driver uses this value to set the AC Line Trigger Slope attribute. See the attribute description for more information	`ViInt32`

Return Values

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

10.4 IviScopeAcLineTrigger Behavior Model

The IviScopeAcLineTrigger group uses the behavior model defined by the IviScopeBase Capabilities.

10.5 IviScopeAcLineTrigger Compliance Notes

IVI Class-Compliant specific drivers that implement this extension group shall implement the AC Line Trigger value for the Trigger Type attribute in the IviScopeBase capabilities group.

11. IviScopeWaveformMeasurement Extension Group

11.1 IviScopeWaveformMeasurement Overview

The IviScopeWaveformMeasurement extension group defines extensions for oscilloscopes capable of calculating various measurements such as rise-time, fall-time, period, and frequency from an acquired waveform.

Table 11-1. Waveform Measurement Descriptions

Name	Description		
		Language	**Identifier**
 Rise Time	The length of time for a rising edge of the signal to rise from the low reference level to the high reference level. The units are seconds.		
		C	IVISCOPE_VAL_RISE_TIME
		COM	IviScopeMeasurementRiseTime
 Fall Time	The length of time for a falling edge of the signal to fall from the high reference level to the low reference level. The units are seconds.		
		C	IVISCOPE_VAL_FALL_TIME
		COM	IviScopeMeasurementFallTime
 Frequency	The frequency of one complete cycle in the waveform. The units are hertz.		
		C	IVISCOPE_VAL_FREQUENCY
		COM	IviScopeMeasurementFrequency
 Period	The length of time of one complete cycle in the waveform. The units are seconds.		
		C	IVISCOPE_VAL_PERIOD
		COM	IviScopeMeasurementPeriod
 Voltage Rms	The true Root Mean Square voltage of the entire waveform. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_RMS
		COM	IviScopeMeasurementVoltageRms
 Voltage Cycle RMS	The true Root Mean Square voltage over an integer number of cycles in the waveform. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_CYCLE_RMS
		COM	IviScopeMeasurementVoltageCycleRms
 Voltage Max	The maximum amplitude found in the entire waveform. The units are volts		
		C	IVISCOPE_VAL_VOLTAGE_MAX
		COM	IviScopeMeasurementVoltageMax
 Voltage Min	The minimum amplitude found in the entire waveform. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_MIN
		COM	IviScopeMeasurementVoltageMin
 Voltage Peak To Peak	The absolute difference between the VOLTAGE_MAX and the VOLTAGE_MIN. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_PEAK_TO_PEAK
		COM	IviScopeMeasurementVoltagePeakToPeak

Table 11-1. Waveform Measurement Descriptions

Name	Description		
		Language	**Identifier**
 Voltage High	The voltage that corresponds to 100% when using the reference levels. The oscilloscope calculates this value using either the min/max or histogram methods. The min/max method uses the maximum value found. The histogram method uses a common value found above the middle of the waveform. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_HIGH
		COM	IviScopeMeasurementVoltageHigh
 Voltage Low	The voltage that corresponds to 0% when using the reference levels. The oscilloscope calculates this value using either the min/max or histogram methods. The min/max method uses the minimum value found. The histogram method uses a common value found below the middle of the waveform. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_LOW
		COM	IviScopeMeasurementVoltageLow
 Voltage Average	The arithmetic average in volts measured over the entire waveform. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_AVERAGE
		COM	IviScopeMeasurementVoltageAverage
 Voltage Cycle Average	The arithmetic average in volts over an integer number of cycles in the waveform. The units are volts.		
		C	IVISCOPE_VAL_VOLTAGE_CYCLE_AVERAGE
		COM	IviScopeMeasurementVoltageCycleAverage
 Width Negative	The length of time between the mid reference level points of a negative pulse in the waveform. The units are seconds.		
		C	IVISCOPE_VAL_WIDTH_NEG
		COM	IviScopeMeasurementWidthNeg
 Width Positive	The length of time between the mid reference level points of a positive pulse in the waveform. The units are seconds.		
		C	IVISCOPE_VAL_WIDTH_POS
		COM	IviScopeMeasurementWidthPos
 Duty Cycle Negative	The ratio of the WIDTH_NEG to the PERIOD of an integer number of cycles in the waveform expressed as a percentage. $DUTY\_CYCLE\_NEG = \frac{WIDTH\_NEG}{PERIOD} \times 100\%$		
		C	IVISCOPE_VAL_DUTY_CYCLE_NEG
		COM	IviScopeMeasurementDutyCycleNeg
 Duty Cycle Positive	The ratio of the WIDTH_POS width to the PERIOD of an integer number of cycles in the waveform expressed as a percentage. $DUTY\_CYCLE\_POS = \frac{WIDTH\_POS}{PERIOD} \times 100\%$		

Table 11-1. Waveform Measurement Descriptions

Name	Description		
		Language	**Identifier**
		C	IVISCOPE_VAL_DUTY_CYCLE_POS
		COM	IviScopeMeasurementDutyCyclePos
 Amplitude	The VOLTAGE_HIGH less the VOLTAGE_LOW in volts over the entire waveform $AMPLITUDE = VOLTAGE\_HIGH - VOLTAGE\_LOW$		
		C	IVISCOPE_VAL_AMPLITUDE
		COM	IviScopeMeasurementAmplitude
 Overshoot	The relative waveform distortion that follows an edge transition. It is calculated using the following formula: -for the rising edge $OVERSHOOT = \frac{local\ maximum - VOLTAGE\_HIGH}{AMPLITUDE} \times 100\%$ where the local maximum is the maximum voltage of the signal in the first half of the time period that commences when the rising edge crosses the high reference level and concludes when the subsequent falling edge crosses the high reference level. -for the falling edge: $OVERSHOOT = \frac{VOLTAGE\_LOW - local\ minimum}{AMPLITUDE} \times 100\%$ where the local minimum is the minimum value of the signal measured in the first half of the time period that commences when the falling edge crosses the low reference level and concludes when the subsequent rising edge crosses the low reference level. The instrument makes the measurement on the edge closest to the beginning of the waveform record. The units are the percentage of the signal amplitude.		
		C	IVISCOPE_VAL_OVERSHOOT
		COM	IviScopeMeasurementOvershoot

Table 11-1. Waveform Measurement Descriptions

Name	Description		
		Language	**Identifier**
 Preshoot	The relative waveform distortion that precedes an edge transition. It is calculated using the following formula: -for the rising edge $\text{PRESHOOT} = \frac{\text{VOLTAGE\_LOW - local minimum}}{\text{AMPLITUDE}} \times 100\%$ where the local minimum is the minimum value of the signal measured in the second half of the time period that commences when the preceding falling edge crosses the low reference level and concludes when the rising edge crosses the low reference level. -for the falling edge: $\text{PRESHOOT} = \frac{\text{local maximum - VOLTAGE\_HIGH}}{\text{AMPLITUDE}} \times 100\%$ where the local maximum is the maximum voltage of the signal in the second half of the time period that commences when the preceding rising edge crosses the high reference level and concludes when the falling edge crosses the high reference level. The instrument makes the measurement on the edge closest to the beginning of the waveform record. The units are the percentage of the signal amplitude.		
		C	IVISCOPE_VAL_PRESHOOT
		COM	IviScopeMeasurementPreshoot

11.2 IviScopeWaveformMeasurement Attributes

The IviScopeWaveformMeasurement capability group defines the following attributes:

- Measurement High Reference
- Measurement Low Reference
- Measurement Middle Reference

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

11.2.1 Measurement High Reference

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

COM Property Name

ReferenceLevel.High

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_MEAS_HIGH_REF

Description

Specifies the high reference the oscilloscope uses for waveform measurements. The value is a percentage of the difference between the Voltage High and Voltage Low.

Compliance Notes

Since measurements depend on this value, the driver shall not perform any coercion of this value, but report an invalid configuration error if the instrument cannot be programmed to a desired value.

11.2.2 Measurement Low Reference

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

COM Property Name

ReferenceLevel.Low

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_MEAS_LOW_REF

Description

Specifies the low reference the oscilloscope uses for waveform measurements. The value is a percentage of the difference between the Voltage High and Voltage Low.

Compliance Notes

Since measurements depend on this value, the driver shall not perform any coercion of this value, but report an invalid configuration error if the instrument cannot be programmed to a desired value.

11.2.3 Measurement Middle Reference

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

COM Property Name

ReferenceLevel.Mid

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_MEAS_MID_REF

Description

Specifies the middle reference the oscilloscope uses for waveform measurements. The value is a percentage of the difference between the Voltage High and Voltage Low.

Compliance Notes

Since measurements depend on this value, the driver shall not perform any coercion of this value, but report an invalid configuration error if the instrument cannot be programmed to a desired value.

11.3 IviScopeWaveformMeasurement Functions

The IviScopeWaveformMeasurement capability group defines the following functions:

- Configure Reference Levels
- Fetch Waveform Measurement
- Read Waveform Measurement

This section describes the behavior and requirements of each function.

11.3.1 Configure Reference Levels

Description

This function configures the reference levels for waveform measurements. Call this function before calling the Read Waveform Measurement or Fetch Waveform Measurement to take waveform measurements.

COM Method Prototype

```
HRESULT ReferenceLevel.Configure ([in] DOUBLE Low,
                                  [in] DOUBLE Mid,
                                  [in] DOUBLE High);
```

C Prototype

```
ViStatus IviScope_ConfigureRefLevels (ViSession Vi,
                                      ViReal64 Low,
                                      ViReal64 Mid,
                                      ViReal64 High);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Low	Measurement low reference. The driver uses this value to set the Meas Low Ref attribute. See the attribute description for more information.	ViReal64
Mid	Measurement mid reference. The driver uses this value to set the Meas Mid Ref attribute. See the attribute description for more information.	ViReal64
High	Measurement high reference. The driver uses this value to set the Meas High Ref attribute. See the attribute description for more information.	ViReal64

Return Values

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

11.3.2 Fetch Waveform Measurement

Description

This function fetches a specified waveform measurement from a specific channel from a previously initiated waveform acquisition. If the channel is not enabled for the acquisition, this function returns the Channel Not Enabled error.

This function obtains a waveform measurement and returns the measurement value. The end-user specifies a particular measurement type, such as rise time, frequency, and voltage peak-to-peak. The waveform on which the oscilloscope calculates the waveform measurement is from an acquisition that was previously initiated.

Use the Initiate Acquisition function to start an acquisition on the channels that were enabled with the Configure Channel function. The oscilloscope acquires waveforms for the enabled channels concurrently. Use the Acquisition Status function to determine when the acquisition is complete. Call this function separately for each waveform measurement on a specific channel.

The end-user can call the Read Waveform Measurement function instead of the Initiate Acquisition function. The Read Waveform Measurement function starts an acquisition on all enabled channels. It then waits for the acquisition to complete, obtains a waveform measurement on the specified channel, and returns the measurement value. Call this function separately to obtain any other waveform measurements on a specific channel.

Configure the appropriate reference levels before calling this function to take a rise time, fall time, width negative, width positive, duty cycle negative, or duty cycle positive measurement.

The end-user can configure the low, mid, and high references either by calling the Configure Reference Levels function or by setting the following attributes.

- Measurement High Reference
- Measurement Low Reference
- Measurement Mid Reference

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

COM Enumeration Name

```
IviScopeMeasurementEnum
```

COM Method Prototype

```
HRESULT Measurements.Item().FetchWaveformMeasurement (
                                        [in] IviScopeMeasurementEnum MeasFunction,
                                        [in,out] DOUBLE Measurement);
```

C Prototype

```
ViStatus IviScope_FetchWaveformMeasurement (ViSession Vi,
                                            ViConstString Channel
                                            ViInt32 MeasFunction,
                                            ViReal64 *Measurement);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Channel	Name of the channel from which to read a waveform.	ViConstString
MeasFunction	Characteristic of the acquired waveform to be measured.	ViInt32

Outputs	Description	Base Type
Measurement	The measured value. The units depend on the measurement that the user specifies with the measFunction parameter.	ViReal64

Defined Values for Measurement Function

Refer to Table 11-1. Waveform Measurement Descriptions for the list of defined values for the MeasFunction parameter.

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
Unable To Perform Measurement	Error: Unable to perform desired waveform measurement operation.
Channel Not Enabled	Error: Specified channel is not enabled for acquisition.

Compliance Notes

1. If an IVI-C class driver defines additional values for the measFunction parameter, the actual values shall be greater than or equal to IVISCOPE_VAL_MEASUREMENT_FUNCTION_CLASS_EXT_BASE and less than IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE.
2. If an IVI-C specific driver defines additional values for this parameter, the actual values shall be greater than or equal to IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE.
3. If an IVI-COM specific driver implements this method with additional values for the measFunction parameter in its instrument specific interfaces, the actual values of the additional elements shall be greater than or equal to Measurement Function Specific Ext Base.

See Section 21, *IviScope Function Parameter Value Definitions*, for the definitions of Measurement Function Specific Ext Base, IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE and IVISCOPE_VAL_MEASUREMENT_FUNCTION_CLASS_EXT_BASE.

11.3.3 Read Waveform Measurement

Description

This function initiates a new waveform acquisition and returns a specified waveform measurement from a specific channel.

This function initiates an acquisition on the channels that the end-user enables with the Configure Channel function. If the channel is not enabled for the acquisition, this function returns Channel Not Enabled error. It then waits for the acquisition to complete, obtains a waveform measurement on the channel the end-user specifies, and returns the measurement value. The end-user specifies a particular measurement type, such as rise time, frequency, and voltage peak-to-peak.

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

The end-user can call the Fetch Waveform Measurement function separately to obtain any other waveform measurement on a specific channel without initiating another acquisition.

The end-user must configure the appropriate reference levels before calling this function. Configure the low, mid, and high references either by calling the Configure Reference Levels function or by setting the following attributes.

- Measurement High Reference
- Measurement Low Reference
- Measurement Middle Reference

COM Enumeration Name

```
IviScopeMeasurementEnum
```

COM Method Prototype

```
HRESULT Measurements.Item().ReadWaveformMeasurement (
                                        [in] IviScopeMeasurementEnum MeasFunction,
                                        [in] LONG MaxTimeMilliseconds,
                                        [in,out] DOUBLE Measurement);
```

C Prototype

```
ViStatus IviScope_ReadWaveformMeasurement (ViSession Vi,
                                           ViConstString Channel
                                           ViInt32 MeasFunction,
                                           ViInt32 MaxTimeMilliseconds,
                                           ViReal64 *Measurement);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Channel	Name of the channel from which to read a waveform.	ViConstString
MeasFunction	Characteristic of the acquired waveform to be measured.	ViInt32
MaxTime Milliseconds	Specifies the maximum time the end-user allows for this function to complete in milliseconds. Defined values: Max Time Immediate - The function returns immediately. If no valid measurement value exists, the function returns an error. Max Time Infinite - The function waits indefinitely for the measurement to complete.	ViInt32

Outputs	Description	Base Type
Measurement	The measured value. The units depend on the measurement that the user specifies with the measFunction parameter.	ViReal64

Defined Values for the MaxTimeMilliseconds Parameter

Name	*Description*		
		Language	*Identifier*
Max Time Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		C	IVISCOPE_VAL_MAX_TIME_IMMEDIATE
		COM	IviScopeTimeOutImmediate
Max Time Infinite	The function waits indefinitely for the measurement to complete.		
		C	IVISCOPE_VAL_MAX_TIME_INFINITE
		COM	IviScopeTimeOutInfinite

Defined Values for Measurement Function

Refer to Table 11-1. Waveform Measurement Descriptions for the list of defined values for the MeasFunction parameter.

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
Channel Not Enabled	Error: Specified channel is not enabled for acquisition.
Max Time Exceeded	Error: Maximum time exceeded before the operation completed.
Unable To Perform Measurement	Error: Unable to perform desired waveform measurement operation.

Compliance Notes

1. The specific instrument driver is not required to support any of the defined values for the `MaxTimeMilliseconds` parameter.
2. If an IVI-C class driver defines additional values for the `measFunction` parameter, the actual values shall be greater than or equal to `IVISCOPE_VAL_MEASUREMENT_FUNCTION_CLASS_EXT_BASE` and less than `IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE`.
3. If an IVI-C specific driver defines additional values for this parameter, the actual values shall be greater than or equal to `IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE`.
4. If an IVI-COM specific driver implements this method with additional values for the `measFunction` parameter in its instrument specific interfaces, the actual values of the additional elements shall be greater than or equal to Measurement Function Specific Ext Base.

See Section 21, *IviScope Function Parameter Value Definitions*, for the definitions of Measurement Function Specific Ext Base, `IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE` and `IVISCOPE_VAL_MEASUREMENT_FUNCTION_CLASS_EXT_BASE`.

11.4 IviScopeWaveformMeasurement Behavior Model

The IviScopeWaveformMeasurement group uses the behavior model defined by the IviScopeBase Capabilities.

12. IviScopeMinMaxWaveform Extension Group

12.1 IviScopeMinMaxWaveform Overview

The IviScopeMinMaxWaveform extension group provides support for oscilloscopes that can acquire minimum and maximum waveforms that correspond to the same range of time. The two most common acquisition types in which oscilloscopes return minimum and maximum waveforms are envelope and peak detect.

12.2 IviScopeMinMaxWaveform Attributes

The IviScopeMinMaxWaveform capability group defines the following attribute:

- Number of Envelopes

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

12.2.1 Number of Envelopes

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Number of Envelopes (IVI-C only)

COM Property Name

Acquisition.NumberOfEnvelopes

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_NUM_ENVELOPES

Description

When the end-user sets the Acquisition Type attribute to Envelope, the oscilloscope acquires multiple waveforms. After each waveform acquisition, the oscilloscope keeps the minimum and maximum values it finds for each point in the waveform record. This attribute specifies the number of waveforms the oscilloscope acquires and analyzes to create the minimum and maximum waveforms. After the oscilloscope acquires as many waveforms as this attribute specifies, it returns to the idle state. This attribute affects instrument operation only when the Acquisition Type attribute is set to Envelope.

12.3 IviScopeMinMaxWaveform Functions

The IviScopeMinMaxWaveform capability group defines the following functions:

- Configure Number of Envelopes
- Fetch Min Max Waveform
- Read Min Max Waveform

This section describes the behavior and requirements of each function.

12.3.1 Configure Number of Envelopes

Description

This function configures the number of waveforms the oscilloscope acquires and analyzes to create the minimum and maximum waveforms.

When the acquisition type is set to Envelope, the oscilloscope acquires multiple waveforms. After each waveform acquisition, the oscilloscope keeps the minimum and maximum values it finds for each element in the waveform record. This function configures the number of waveforms the oscilloscope acquires and analyzes to create the minimum and maximum waveforms.

After the oscilloscope acquires the specified number of waveforms, it returns to the idle state.

Set the acquisition type to Envelope before calling this function.

COM Method Prototype

`N/A`
(use the `Acquisition.NumEnvelopes` property)

C Prototype

```
ViStatus IviScope_ConfigureNumEnvelopes (ViSession Vi,
                                         ViInt32 NumEnvelopes);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`NumEnvelopes`	Specifies the number of waveforms the oscilloscope acquires and analyzes to create the minimum and maximum waveforms. The driver sets the Number of Envelopes attribute to this value. See the Number of Envelopes attribute for a complete description.	`ViInt32`

Return Values

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

12.3.2 Fetch Min Max Waveform

Description

This function returns the minimum and maximum waveforms that the oscilloscope acquires for the specified channel. If the channel is not enabled for the acquisition, this function returns the Channel Not Enabled error.

The waveforms are from a previously initiated acquisition. Use this function to fetch waveforms when the acquisition type is set to Peak Detect or Envelope. If the acquisition type is not one of the listed types, the function returns the Invalid Acquisition Type error.

Use the Initiate Acquisition function to start an acquisition on the enabled channels. The oscilloscope acquires the min/max waveforms for the enabled channels concurrently. Use the Acquisition Status function to determine when the acquisition is complete. The end-user must call this function separately for each enabled channel to obtain the min/max waveforms.

The end-user can call the Read Min Max Waveform function instead of the Initiate Acquisition function. The Read Min Max Waveform function starts an acquisition on all enabled channels, waits for the acquisition to complete, and returns the min/max waveforms for the specified channel. You call this function to obtain the min/max waveforms for each of the remaining channels.

After this function executes, each element in the `MinWaveform` and `MaxWaveform` parameters is either a voltage or a value indicating that the oscilloscope could not sample a voltage.

The C end-user configures the interpolation method the oscilloscope uses with the `IviScope_ConfigureInterpolation` function. The COM end-user uses the Acquisition.Interpolation property. If interpolation is disabled, the oscilloscope does not interpolate points in the waveform. If the oscilloscope cannot sample a value for a point in the waveform, the driver sets the corresponding element in the `MinWaveform` or `MaxWaveform` to an IEEE-defined NaN (Not a Number) value and the function returns the Invalid Waveform Element warning.

Use the Is Waveform Element Invalid function to test each element in the `MinWaveform` and `MaxWaveform` parameters for an invalid waveform element.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

COM Method Prototype

```
HRESULT Measurements.Item().FetchWaveformMinMax (
                                    [in,out] SAFEARRAY(DOUBLE) *MinWaveform,
                                    [in,out] SAFEARRAY(DOUBLE) *MaxWaveform,
                                    [in,out] DOUBLE *InitialX,
                                    [in,out] DOUBLE *XIncrement);
```

C Prototype

```
ViStatus IviScope_FetchMinMaxWaveform (ViSession Vi,
                                       ViConstString Channel,
                                       ViInt32 WaveformSize,
                                       ViReal64 MinWaveform[],
                                       ViReal64 MaxWaveform[],
                                       ViInt32 *ActualPoints,
                                       ViReal64 *InitialX,
                                       ViReal64 *XIncrement);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Channel	Name of the channel from which to read a waveform.	ViConstString
WaveformSize	Specifies the number of elements in the MinWaveform and MaxWaveform arrays.	ViInt32

Outputs	Description	Base Type
MinWaveform	A user-allocated buffer into which the min waveform is stored.	ViReal64[]
MaxWaveform	A user-allocated buffer into which the max waveform is stored.	ViReal64[]
ActualPoints	Number of points actually acquired in each waveform	ViInt32
InitialX	The time in relation to the Trigger Event of the first point in the waveform in seconds.	ViReal64
XIncrement	The time between points in the acquired waveform in seconds.	ViReal64

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 Waveform Element	Warning: One of the elements in the waveform array is invalid.
Channel Not Enabled	Error: Specified channel is not enabled for acquisition.
Invalid Acquisition Type	Error: Invalid acquisition type.

12.3.3 Read Min Max Waveform

Description

This function initiates new waveform acquisition and returns minimum and maximum waveforms from a specific channel. If the channel is not enabled for the acquisition, this function returns the Channel Not Enabled error.

This function is used when the Acquisition Type is Peak Detect or Envelope . If the acquisition type is not one of the listed types, the function returns the Invalid Acquisition Type error.

This function initiates an acquisition on the enabled channels. It then waits for the acquisition to complete, and returns the min/max waveforms for the specified channel. Call the Fetch Min Max Waveform function to obtain the min/max waveforms for each of the remaining enabled channels without initiating another acquisition. If the oscilloscope did not complete the acquisition within the time period the user specified with the `MaxTimeMilliseconds` parameter, the function returns the Max Time Exceeded error.

The C end-user configures the interpolation method the oscilloscope uses with the `IviScope_ConfigureInterpolation` function. The COM end-user uses the Acquisition.Interpolation property.If interpolation is disabled, the oscilloscope does not interpolate points in the waveform. If the oscilloscope cannot sample a value for a point in the waveform, the driver sets the corresponding element in the `MinWaveform or MaxWaveform` to an IEEE-defined NaN (Not a Number) value and the function returns the Invalid Waveform Element warning.

Use the Is Waveform Element Invalid function to test each element in the `MinWaveform` and `MaxWaveform a`rray parameters for an invalid waveform element.

COM Method Prototype

```
HRESULT Measurements.Item().ReadWaveformMinMax (
                                    [in] LONG MaxTimeMilliseconds,
                                    [in,out] SAFEARRAY(DOUBLE) *MinWaveform,
                                    [in,out] SAFEARRAY(DOUBLE) *MaxWaveform,
                                    [in,out] DOUBLE *InitialX,
                                    [in,out] DOUBLE *XIncrement);
```

C Prototype

```
ViStatus IviScope_ReadMinMaxWaveform (ViSession Vi,
                                      ViConstString Channel,
                                      ViInt32 WaveformSize,
                                      ViInt32 MaxTimeMilliseconds,
                                      ViReal64 MinWaveform[],
                                      ViReal64 MaxWaveform[],
                                      ViInt32 *ActualPoints,
                                      ViReal64 *InitialX,
                                      ViReal64 *XIncrement);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession
Channel	Name of the channel from which to read a waveform.	ViConstString
WaveformSize	Specifies the number of elements in the minWaveform and maxWaveform arrays.	ViInt32
MaxTime Milliseconds	Specifies the maximum time allowed for this function to complete in milliseconds. Defined values: Max Time Immediate - The function returns immediately. If no valid measurement value exists, the function returns an error. Max Time Infinite - The function waits indefinitely for the measurement to complete.	ViInt32

Outputs	Description	Base Type
MinWaveform	A user-allocated buffer into which the min waveform is stored.	ViReal64[]
MaxWaveform	A user-allocated buffer into which the max waveform is stored.	ViReal64[]
ActualPoints	Number of points actually acquired in each waveform.	ViInt32
InitialX	The time in relation to the Trigger Event of the first point in the waveform in seconds.	ViReal64
XIncrement	The time between points in the acquired waveform in seconds.	ViReal64

Defined Values for the MaxTimeMilliseconds Parameter

Name	*Description*		
		Language	*Identifier*
Max Time Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		C	IVISCOPE_VAL_MAX_TIME_IMMEDIATE
		COM	IviScopeTimeOutImmediate
Max Time Infinite	The function waits indefinitely for the measurement to complete.		
		C	IVISCOPE_VAL_MAX_TIME_INFINITE
		COM	IviScopeTimeOutInfinite

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 Waveform Element	Warning: One of the elements in the waveform array is invalid.
Channel Not Enabled	Error: Specified channel is not enabled for acquisition.
Invalid Acquisition Type	Error: Invalid acquisition type.
Max Time Exceeded	Error: Maximum time exceeded before the operation completed.

Compliance Notes

The specific instrument driver is not required to support any of the defined values for the `MaxTimeMilliseconds` parameter.

12.4 IviScopeMinMaxWaveform Behavior Model

The IviScopeMinMaxWaveform group uses the behavior model defined by the IviScopeBase Capabilities.

12.5 IviScopeMinMaxWaveform Compliance Notes

1. IVI Class-Compliant specific drivers that implement this extension group shall implement at least one of the following values for the Acquisition Type attribute in the IviScopeBase capabilities group:
   - Peak Detect
   - Envelope
2. An IVI Class-Compliant specific driver does not have to implement the Number of Envelopes attribute if it does not implement the Envelope value for the Acquisition Type attribute. In addition, an IVI-C specific driver does not have to implement the Configure Number of Envelopes function if it does not implement the Envelope value for the Acquisition Type attribute.

13. IviScopeProbeAutoSense Extension Group

13.1 IviScopeProbeAutoSense Overview

The IviScopeProbeAutoSense extension group provides support for oscilloscopes that can return the probe attenuation of the attached probe.

13.2 IviScopeProbeAutoSense Attributes

The IviScopeProbeAutoSense capability group defines the following attribute:

- Probe Sense Value

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

13.2.1 Probe Sense Value

Data Type	Access	Applies to	Coercion	High Level Functions
ViReal64	RO	Channels	None	Auto Probe Sense Value (IVI-C only)

COM Property Name

Channels.Item().ProbeSense

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_PROBE_SENSE_VALUE

Description

Returns the probe attenuation value that the oscilloscope automatically senses. This value is the scaling factor by which the probe the end-user attaches to the channel attenuates the input. If the auto probe sense capability is not enabled, this attribute returns the current manual probe attenuation setting.

13.3 IviScopeProbeAutoSense Functions

The IviScopeProbeAutoSense capability group defines the following function:

- Auto Probe Sense Value (IVI-C only)

This section describes the behavior and requirements of this function.

13.3.1 Auto Probe Sense Value (IVI-C only)

Description

The function returns the probe attenuation value the oscilloscope senses. The capability is enabled by setting the `probeAttenuation` parameter of the Configure Channel function to Probe Sense On.

If the automatic probe sense capability is disabled, this function returns the manual probe attenuation setting.

COM Method Prototype

`N/A`
(use the `Channels.Item().ProbeSense` property)

C Prototype

```
ViStatus IviScope_AutoProbeSenseValue (ViSession Vi,
                                       ViConstString Channel,
                                       ViReal64 *AutoProbeSenseValue);
```

Parameters

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

Outputs	Description	Base Type
`AutoProbeSense Value`	Returns the probe attenuation value the oscilloscope senses. The driver returns the value of the Probe Sense Value attribute. See the Probe Sense Value attribute for a complete description.	`ViReal64`

Return Values

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

13.4 IviScopeProbeAutoSense Behavior Model

The IviScopeProbeAutoSense group uses the behavior model defined by the IviScopeBase Capabilities.

13.5 IviScopeProbeAutoSense Compliance Notes

IVI Class-Compliant specific drivers that implement this extension group shall implement the Probe Sense On value for the Probe Attenuation attribute in the IviScopeBase capabilities group.

14. IviScopeContinuousAcquisition Extension Group

14.1 IviScopeContinuousAcquisition Overview

The IviScopeContinuousAcquisition extension group provides support for oscilloscopes that can perform a continuous acquisition.

14.2 IviScopeContinuousAcquisition Attributes

The IviScopeContinuousAcquisition capability group defines the following attribute:

- Initiate Continuous

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

14.2.1 Initiate Continuous

Data Type	Access	Applies to	Coercion	High Level Functions
ViBoolean	R/W	N/A	None	Configure Initiate Continuous (IVI-C only)

COM Property Name

Trigger.Continuous

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_INITIATE_CONTINUOUS

Description

Specifies whether the oscilloscope continuously initiates waveform acquisition. If the end-user sets this attribute to True, the oscilloscope immediately waits for another trigger after the previous waveform acquisition is complete. Setting this attribute to True is useful when the end-user requires continuous updates of the oscilloscope display. This specification does not define the behavior of the read waveform and fetch waveform functions when this attribute is set to True. The behavior of these functions is instrument specific.

Defined Values

Name	*Description*		
		Language	*Identifier*
True	The oscilloscope continuously acquires a waveform		
		C	VI_TRUE
		COM	True
False	The oscilloscope does not acquire a waveform continuously		
		C	VI_FALSE
		COM	False

Compliance Notes

The IVI Class-Compliant specific driver shall implement both the True and False values.

14.3 IviScopeContinuousAcquisition Functions

The IviScopeContinuousAcquisition capability group defines the following function:

- Configure Initiate Continuous (IVI-C only)

This section describes the behavior and requirements of this function.

14.3.1 Configure Initiate Continuous (IVI-C only)

Description

This function configures the oscilloscope to perform a continuous acquisition.

COM Method Prototype

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

C Prototype

```
ViStatus IviScope_ConfigureInitiateContinuous (ViSession Vi,
                                               ViBoolean ContinuousAcquisition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Continuous Acquisition`	Specifies if the oscilloscope is enabled for continuous acquisition. The driver uses this value to set the Initiate Continuous attribute. See the attribute description for more information	`ViBoolean`

Return Values

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

14.4 IviScopeContinuousAcquisition Behavior Model

The following behavior diagram shows relationships between IviScopeContinuousAcquisition Capabilities and oscilloscope behavior.

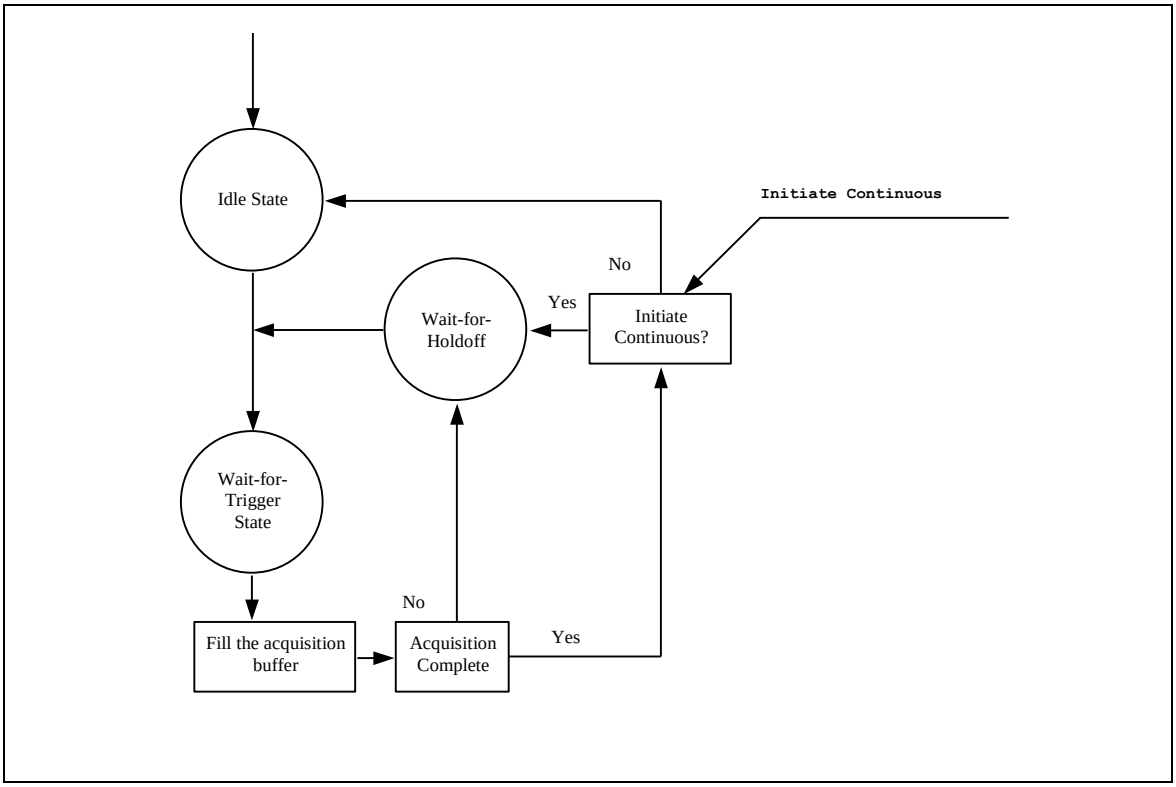


Figure 14-1. IviScopeContinuousAcquisition Behavior Model

The IviScopeContinuousAcquisition extension group adds the attribute that controls whether the instrument operates in a single-shot mode or if it acquires the data continuously.

After the oscilloscope completes an acquisition, if the Initiate Continuous attribute is set to True, the instrument goes to the *Wait-for-Trigger* state instead of returning to the *Idle* state. Setting this attribute to True is useful when the end-user requires continuous updates of the oscilloscope display. This specification does not define the behavior of the read and fetch functions when this attribute is set to True. The behavior of these functions is instrument specific.

15. IviScopeAverageAcquisition Extension Group

15.1 IviScopeAverageAcquisition Overview

The IviScopeAverageAcquisition extension group provides support for oscilloscopes that can perform the average acquisition.

15.2 IviScopeAverageAcquisition Attributes

The IviScopeAverageAcquisition capability group defines the following attribute:

- Number of Averages

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

15.2.1 Number of Averages

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	None	Configure Number of Averages (IVI-C only)

COM Property Name

Acquisition.NumberOfAverages

COM Enumeration Name

N/A

C Constant Name

IVISCOPE_ATTR_NUM_AVERAGES

Description

Specifies the number of waveform the oscilloscope acquires and averages. After the oscilloscope acquires as many waveforms as this attribute specifies, it returns to the idle state. This attribute affects instrument behavior only when the Acquisition Type attribute is set to Average.

15.3 IviScopeAverageAcquisition Functions

The IviScopeAverageAcquisition capability group defines the following function:

- Configure Number of Averages (IVI-C only)

This section describes the behavior and requirements of this function.

15.3.1 Configure Number of Averages (IVI-C only)

Description

This function configures the number of waveforms that the oscilloscope acquires and averages. After the oscilloscope acquires number of waveforms specified, it returns to the idle state.

Set the acquisition type to Average before calling this function. If the acquisition type is not set to Average, the function returns the Invalid Acquisition Type error.

COM Method Prototype

`N/A`
(use the `Acquisition.NumberOfAverages` property)

C Prototype

```
ViStatus IviScope_ConfigureNumberOfAverages (ViSession Vi,
                                             ViInt32 NumberOfAverages);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`NumberOfAverages`	Specifies the number of waveforms the oscilloscope acquires and averages. The driver sets the Number of Averages attribute to this value.	`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
Invalid Acquisition Type	Error: Invalid acquisition type.

15.4 IviScopeAverageAcquisition Behavior Model

The IviScopeAverageAcquisition group uses the behavior model defined by the IviScopeBase Capabilities.

15.5 IviScopeAverageAcquisition Compliance Notes

IVI Class-Compliant specific drivers that implement this extension group shall implement the Average value for the Acquisition Type attribute in the IviScopeBase capabilities group.

16. IviScopeSampleMode Extension Group

16.1 IviScopeSampleMode Overview

The IviScopeSampleMode extension group provides support for oscilloscopes that can return whether they are using equivalent time or real time sampling to acquire waveform.

16.2 IviScopeSampleMode Attributes

The IviScopeSampleMode capability group defines the following attribute:

- Sample Mode

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

16.2.1 Sample Mode

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

COM Property Name

Acquisition.SampleMode

COM Enumeration Name

IviScopeSampleModeEnum

C Constant Name

IVISCOPE_ATTR_SAMPLE_MODE

Description

Returns the sample mode the oscilloscope is currently using.

Defined Values

Name	*Description*		
		Language	*Identifier*
Real Time	Indicates that the oscilloscope is using real-time sampling.		
		C	IVISCOPE_VAL_REAL_TIME
		COM	IviScopeSampleModeRealTime
Equivalent Time	Indicates that the oscilloscope is using equivalent time sampling.		
		C	IVISCOPE_VAL_EQUIVALENT_TIME
		COM	IviScopeSampleModeEquivalentTime

16.3 IviScopeSampleMode Functions

The IviScopeSampleMode capability group defines the following function:

- Sample Mode (IVI-C only)

This section describes the behavior and requirements of this function.

16.3.1 Sample Mode (IVI-C only)

Description

This function returns the sample mode the oscilloscope is currently using.

COM Method Prototype

`N/A`
(use the `Acquisition.SampleMode` property)

C Prototype

```
ViStatus IviScope_SampleMode (ViSession Vi,
                              ViInt32 *SampleMode);
```

Parameters

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

Outputs	Description	Base Type
`SampleMode`	Returns the sample mode the oscilloscope is currently using. The driver returns the value of the Sample Mode attribute. See the Sample Mode attribute for a complete description and defined values.	`ViInt32`

Return Values

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

16.4 IviScopeSampleMode Behavior Model

The IviScopeSampleMode group uses the behavior model defined by the IviScopeBase Capabilities.

17. IviScopeTriggerModifier Extension Group

17.1 IviScopeTriggerModifier Overview

The IviScopeTriggerModifier extension group provides support for oscilloscopes that can specify the behavior of the triggering subsystem in the absence of the configured trigger.

17.2 IviScopeTriggerModifier Attributes

The IviScopeTriggerModifier capability group defines the following attribute:

- Trigger Modifier

This section describes the behavior and requirements of this attribute. The actual value for this attribute ID is defined in Section 19, *IviScope Attribute ID Definitions*.

17.2.1 Trigger Modifier

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	No	None	IviScope_ConfigureTriggerModifier

COM Property Name

Trigger.Modifier

COM Enumeration Name

IviScopeTriggerModifierEnum

C Constant Name

IVISCOPE_ATTR_TRIGGER_MODIFIER

Description

Specifies the trigger modifier. The trigger modifier determines the oscilloscope's behavior in the absence of the configured trigger.

Defined Values

Name	*Description*		
		Language	*Identifier*
No Trigger Modifier	The oscilloscope waits until the trigger the end-user specifies occurs.		
		C	IVISCOPE_VAL_NO_TRIGGER_MOD
		COM	IviScopeTriggerModifierNone
Auto	The oscilloscope automatically triggers if the configured trigger does not occur within the oscilloscope's timeout period.		
		C	IVISCOPE_VAL_AUTO
		COM	IviScopeTriggerModifierAuto
Auto Level	The oscilloscope adjusts the trigger level if the trigger the end-user specifies does not occur.		
		C	IVISCOPE_VAL_AUTO_LEVEL
		COM	IviScopeTriggerModifierAutoLevel

Compliance Notes

1. Instrument driver shall support the value No Trigger Modifier and at least one of the following values:
   - Auto
   - Auto Level
2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TRIGGER_MOD_CLASS_EXT_BASE and less than IVISCOPE_VAL_TRIGGER_MOD_SPECIFIC_EXT_BASE.
3. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to IVISCOPE_VAL_TRIGGER_MOD_SPECIFIC_EXT_BASE.

17.3 IviScopeTriggerModifier Functions

The IviScopeTriggerModifier capability group defines the following function:

- Configure Trigger Modifier (IVI-C only)

This section describes the behavior and requirements of this function.

17.3.1 Configure Trigger Modifier (IVI-C only)

Description

This function configures the oscilloscope's trigger modifier.

COM Method Prototype

`N/A`
(use the `Trigger.Modifier` property)

C Prototype

```
ViStatus IviScope_ConfigureTriggerModifier (ViSession Vi,
                                            ViInt32 TriggerModifier);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`TriggerModifier`	Specifies the method the oscilloscope uses in the absence of trigger conditions. The driver sets the Trigger Modifier attribute to this value. See the attribute description for more information and defined values.	`ViInt32`

Return Values

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

17.4 IviScopeTriggerModifier Behavior Model

The IviScopeTriggerModifier group uses the behavior model defined by the IviScopeBase Capabilities.

18. IviScopeAutoSetup Extension Group

18.1 IviScopeAutoSetup Overview

The IviScopeAutoSetup extension group provides support for oscilloscopes that can perform an auto-setup operation.

18.2 IviScopeAutoSetup Functions

The IviScopeAutoSetup capability group defines the following function:

- Auto Setup

This section describes the behavior and requirements of this function.

18.2.1 Auto Setup

Description

This function performs an auto-setup on the instrument.

COM Method Prototype

```
HRESULT Measurements.AutoSetup()
```

C Prototype

```
ViStatus IviScope_AutoSetup (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle	ViSession

Return Values

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

18.3 IviScopeAutoSetup Behavior Model

The IviScopeAutoSetup group uses the behavior model defined by the IviScopeBase Capabilities.

19. IviScope Attribute ID Definitions

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

Table 19-1. IviScope Attribute ID Values

Attribute Name	ID Value
IVISCOPE_ATTR_VERTICAL_RANGE	IVI_CLASS_ATTR_BASE + 1
IVISCOPE_ATTR_VERTICAL_OFFSET	IVI_CLASS_ATTR_BASE + 2
IVISCOPE_ATTR_VERTICAL_COUPLING	IVI_CLASS_ATTR_BASE + 3
IVISCOPE_ATTR_PROBE_ATTENUATION	IVI_CLASS_ATTR_BASE + 4
IVISCOPE_ATTR_CHANNEL_ENABLED	IVI_CLASS_ATTR_BASE + 5
IVISCOPE_ATTR_MAX_INPUT_FREQUENCY	IVI_CLASS_ATTR_BASE + 6
IVISCOPE_ATTR_HORZ_TIME_PER_RECORD	IVI_CLASS_ATTR_BASE + 7
IVISCOPE_ATTR_HORZ_RECORD_LENGTH	IVI_CLASS_ATTR_BASE + 8
IVISCOPE_ATTR_HORZ_MIN_NUM_PTS	IVI_CLASS_ATTR_BASE + 9
IVISCOPE_ATTR_HORZ_SAMPLE_RATE	IVI_CLASS_ATTR_BASE + 10
IVISCOPE_ATTR_TRIGGER_TYPE	IVI_CLASS_ATTR_BASE + 12
IVISCOPE_ATTR_TRIGGER_SOURCE	IVI_CLASS_ATTR_BASE + 13
IVISCOPE_ATTR_TRIGGER_COUPLING	IVI_CLASS_ATTR_BASE + 14
IVISCOPE_ATTR_TRIGGER_HOLDOFF	IVI_CLASS_ATTR_BASE + 16
IVISCOPE_ATTR_TRIGGER_LEVEL	IVI_CLASS_ATTR_BASE + 17
IVISCOPE_ATTR_TRIGGER_SLOPE	IVI_CLASS_ATTR_BASE + 18
IVISCOPE_ATTR_INTERPOLATION	IVI_CLASS_ATTR_BASE + 19
IVISCOPE_ATTR_ACQUISITION_TYPE	IVI_CLASS_ATTR_BASE + 101
IVISCOPE_ATTR_TRIGGER_MODIFIER	IVI_CLASS_ATTR_BASE + 102
IVISCOPE_ATTR_INPUT_IMPEDANCE	IVI_CLASS_ATTR_BASE + 103
IVISCOPE_ATTR_NUM_AVERAGES	IVI_CLASS_ATTR_BASE + 104
IVISCOPE_ATTR_NUM_ENVELOPES	IVI_CLASS_ATTR_BASE + 105
IVISCOPE_ATTR_SAMPLE_MODE	IVI_CLASS_ATTR_BASE + 106
IVISCOPE_ATTR_INITIATE_CONTINUOUS	IVI_CLASS_ATTR_BASE + 107
IVISCOPE_ATTR_PROBE_SENSE_VALUE	IVI_CLASS_ATTR_BASE + 108
IVISCOPE_ATTR_ACQUISITION_START_TIME	IVI_CLASS_ATTR_BASE + 109
IVISCOPE_ATTR_TV_TRIGGER_SIGNAL_FORMAT	IVI_CLASS_ATTR_BASE + 201
IVISCOPE_ATTR_TV_TRIGGER_POLARITY	IVI_CLASS_ATTR_BASE + 204
IVISCOPE_ATTR_TV_TRIGGER_EVENT	IVI_CLASS_ATTR_BASE + 205
IVISCOPE_ATTR_TV_TRIGGER_LINE_NUMBER	IVI_CLASS_ATTR_BASE + 206
IVISCOPE_ATTR_RUNT_HIGH_THRESHOLD	IVI_CLASS_ATTR_BASE + 301
IVISCOPE_ATTR_RUNT_LOW_THRESHOLD	IVI_CLASS_ATTR_BASE + 302
IVISCOPE_ATTR_RUNT_POLARITY	IVI_CLASS_ATTR_BASE + 303
IVISCOPE_ATTR_GLITCH_WIDTH	IVI_CLASS_ATTR_BASE + 401
IVISCOPE_ATTR_GLITCH_POLARITY	IVI_CLASS_ATTR_BASE + 402
IVISCOPE_ATTR_GLITCH_CONDITION	IVI_CLASS_ATTR_BASE + 403
IVISCOPE_ATTR_WIDTH_LOW_THRESHOLD	IVI_CLASS_ATTR_BASE + 501

Table 19-1. IviScope Attribute ID Values

Attribute Name	ID Value
IVISCOPE_ATTR_WIDTH_HIGH_THRESHOLD	IVI_CLASS_ATTR_BASE + 502
IVISCOPE_ATTR_WIDTH_POLARITY	IVI_CLASS_ATTR_BASE + 503
IVISCOPE_ATTR_WIDTH_CONDITION	IVI_CLASS_ATTR_BASE + 504
IVISCOPE_ATTR_MEAS_HIGH_REF	IVI_CLASS_ATTR_BASE + 607
IVISCOPE_ATTR_MEAS_LOW_REF	IVI_CLASS_ATTR_BASE + 608
IVISCOPE_ATTR_MEAS_MID_REF	IVI_CLASS_ATTR_BASE + 609
IVISCOPE_ATTR_AC_LINE_TRIGGER_SLOPE	IVI_CLASS_ATTR_BASE + 701
IVISCOPE_ATTR_CHANNEL_COUNT	IVI_INHERENT_ATTR_BASE + 203
IVISCOPE_ATTR_MEASUREMENT_CHANNEL_COUNT	IVI_CLASS_ATTR_BASE + 802

19.1 IviScope Obsolete Attribute Names

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

IVISCOPE_ATTR_BANDWIDTH

IVISCOPE_ATTR_HORZ_RECORD_REF_POSITION

IVISCOPE_ATTR_TRIGGER_DELAY_TIME

IVISCOPE_ATTR_TV_TRIGGER_SIGNAL_TYPE

IVISCOPE_ATTR_TV_TRIGGER_FIELD

IVISCOPE_ATTR_TV_TRIGGER_LINE

19.2 IviScope Obsolete Attribute ID Values

The following attribute IDs are reserved by the IviScope specification 1.0. Future versions of this specification cannot use these values:

1250011 (IVI_CLASS_ATTR_BASE + 11)

1250015 (IVI_CLASS_ATTR_BASE + 15)

1250202 (IVI_CLASS_ATTR_BASE + 202)

1250203 (IVI_CLASS_ATTR_BASE + 203)

20. IviScope Attribute Value Definitions

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

Acquisition Type

Value Name	Language	Identifier	Actual Value
Normal	C	IVISCOPE_VAL_NORMAL	0
	COM	IviScopeAcquisitionTypeNormal	0
Peak Detect	C	IVISCOPE_VAL_PEAK_DETECT	1
	COM	IviScopeAcquisitionTypePeakDetect	1
High Resolution	C	IVISCOPE_VAL_HI_RES	2
	COM	IviScopeAcquisitionTypeHiRes	2
Envelope	C	IVISCOPE_VAL_ENVELOPE	3
	COM	IviScopeAcquisitionTypeEnvelope	3
Average	C	IVISCOPE_VAL_AVERAGE	4
	COM	IviScopeAcquisitionTypeAverage	4
Acquisition Type Class Ext Base	C	IVISCOPE_VAL_ACQUISITION_TYPE_CLASS_EXT_BASE	100
Acquisition Type Specific Ext Base	C	IVISCOPE_VAL_ACQUISITION_TYPE_SPECIFIC_EXT_BASE	1000
	COM	N/A	

Probe Attenuation

Value Name	Language	Identifier	Actual Value
Probe Sense On	C	IVISCOPE_VAL_PROBE_SENSE_ON	-1
	COM	-1	-1
Probe Attenuation Class Ext Base	C	IVISCOPE_VAL_PROBE_ATTENUATION_CLASS_EXT_BASE	-100
Probe Attenuation Specific Ext Base	C	IVISCOPE_VAL_PROBE_ATTENUATION_SPECIFIC_EXT_BASE	-1000
	COM	N/A	

Vertical Coupling

Value Name	Language	Identifier	Actual Value
AC	C	IVISCOPE_VAL_AC	0
	COM	IviScopeVerticalCouplingAC	0
DC	C	IVISCOPE_VAL_DC	1
	COM	IviScopeVerticalCouplingDC	1
Gnd	C	IVISCOPE_VAL_GND	2
	COM	IviScopeVerticalCouplingGnd	2
Vertical Coupling Class Ext Base	C	IVISCOPE_VAL_VERTICAL_COUPLING_CLASS_EXT_BASE	100
Vertical Coupling Specific Ext Base	C	IVISCOPE_VAL_VERTICAL_COUPLING_SPECIFIC_EXT_BASE	1000
	COM	N/A	

Trigger Coupling

Value Name	Language	Identifier	Actual Value
AC	C	IVISCOPE_VAL_AC	0
	COM	IviScopeTriggerCouplingAC	0
DC	C	IVISCOPE_VAL_DC	1
	COM	IviScopeTriggerCouplingDC	1
HF Reject	C	IVISCOPE_VAL_HF_REJECT	3
	COM	IviScopeTriggerCouplingHFReject	3
LF Reject	C	IVISCOPE_VAL_LF_REJECT	4
	COM	IviScopeTriggerCouplingLFReject	4
Noise Reject	C	IVISCOPE_VAL_NOISE_REJECT	5
	COM	IviScopeTriggerCouplingNoiseReject	5
Trigger Coupling Class Ext Base	C	IVISCOPE_VAL_TRIGGER_COUPLING_CLASS_EXT_BASE	100
Trigger Coupling Specific Ext Base	C	IVISCOPE_VAL_TRIGGER_COUPLING_SPECIFIC_EXT_BASE	1000
	COM	N/A	

Trigger Slope

Value Name	Language	Identifier	Actual Value
Negative	C	IVISCOPE_VAL_NEGATIVE	0
	COM	IviScopeTriggerSlopeNegative	0
Positive	C	IVISCOPE_VAL_POSITIVE	1
	COM	IviScopeTriggerSlopePositive	1

Trigger Source

Value Name	Language	Identifier	Actual Value
ECL0	C	IVISCOPE_VAL_ECL0	"VAL_ECL0"
	COM	"VAL_ECL0"	"VAL_ECL0"
ECL1	C	IVISCOPE_VAL_ECL1	"VAL_ECL1"
	COM	"VAL_ECL1"	"VAL_ECL1"
External	C	IVISCOPE_VAL_EXTERNAL	"VAL_EXTERNAL"
	COM	"VAL_EXTERNAL"	"VAL_EXTERNAL"
PXI Star	C	IVISCOPE_VAL_PXI_STAR	"VAL_PXI_STAR"
	COM	"VAL_PXI_STAR"	"VAL_PXI_STAR"
RTSI0	C	IVISCOPE_VAL_RTSI_0	"VAL_RTSI_0"
	COM	"VAL_RTSI_0"	"VAL_RTSI_0"
RTSI1	C	IVISCOPE_VAL_RTSI_1	"VAL_RTSI_1"
	COM	"VAL_RTSI_1"	"VAL_RTSI_1"
RTSI2	C	IVISCOPE_VAL_RTSI_2	"VAL_RTSI_2"
	COM	"VAL_RTSI_2"	"VAL_RTSI_2"
RTSI3	C	IVISCOPE_VAL_RTSI_3	"VAL_RTSI_3"
	COM	"VAL_RTSI_3"	"VAL_RTSI_3"
RTSI4	C	IVISCOPE_VAL_RTSI_4	"VAL_RTSI_4"
	COM	"VAL_RTSI_4"	"VAL_RTSI_4"
RTSI5	C	IVISCOPE_VAL_RTSI_5	"VAL_RTSI_5"
	COM	"VAL_RTSI_5"	"VAL_RTSI_5"
RTSI6	C	IVISCOPE_VAL_RTSI_6	"VAL_RTSI_6"
	COM	"VAL_RTSI_6"	"VAL_RTSI_6"
TTL0	C	IVISCOPE_VAL_TTL0	"VAL_TTL0"
	COM	"VAL_TTL0"	"VAL_TTL0"
TTL1	C	IVISCOPE_VAL_TTL1	"VAL_TTL1"
	COM	"VAL_TTL1"	"VAL_TTL1"
TTL2	C	IVISCOPE_VAL_TTL2	"VAL_TTL2"
	COM	"VAL_TTL2"	"VAL_TTL2"
TTL3	C	IVISCOPE_VAL_TTL3	"VAL_TTL3"

	COM	"VAL_TTL3"	"VAL_TTL3"
TTL4	C	IVISCOPE_VAL_TTL4	"VAL_TTL4"
	COM	"VAL_TTL4"	"VAL_TTL4"
TTL5	C	IVISCOPE_VAL_TTL5	"VAL_TTL5"
	COM	"VAL_TTL5"	"VAL_TTL5"
TTL6	C	IVISCOPE_VAL_TTL6	"VAL_TTL6"
	COM	"VAL_TTL6"	"VAL_TTL6"
TTL7	C	IVISCOPE_VAL_TTL7	"VAL_TTL7"
	COM	"VAL_TTL7"	"VAL_TTL7"

The following values are reserved by the IviScope specification 1.0 for the IVISCOPE_ATTR_TRIGGER_SOURCE attribute. Future versions of this specification cannot use these values:

"VAL_IMMEDIATE"

"VAL_GPIB_GET"

"VAL_SW_TRIG_FUNC"

"VAL_AC_LINE"

Trigger Type

Value Name	Language	Identifier	Actual Value
Edge Trigger	C	IVISCOPE_VAL_EDGE_TRIGGER	1
	COM	IviScopeTriggerEdge	1
Width Trigger	C	IVISCOPE_VAL_WIDTH_TRIGGER	2
	COM	IviScopeTriggerWidth	2
Runt Trigger	C	IVISCOPE_VAL_RUNT_TRIGGER	3
	COM	IviScopeTriggerRunt	3
Glitch Trigger	C	IVISCOPE_VAL_GLITCH_TRIGGER	4
	COM	IviScopeTriggerGlitch	4
TV Trigger	C	IVISCOPE_VAL_TV_TRIGGER	5
	COM	IviScopeTriggerTV	5
Immediate Trigger	C	IVISCOPE_VAL_IMMEDIATE_TRIGGER	6
	COM	IviScopeTriggerImmediate	6
AC Line Trigger	C	IVISCOPE_VAL_AC_LINE_TRIGGER	7
	COM	IviScopeTriggerACLine	7
Trigger Type Class Ext Base	C	IVISCOPE_VAL_TRIGGER_TYPE_ CLASS_EXT_BASE	200
Trigger Type Specific Ext Base	C	IVISCOPE_VAL_TRIGGER_TYPE_ SPECIFIC_EXT_BASE	1000
	COM	N/A	

The following values are reserved by the IviScope specification 1.0 for the IVISCOPE_ATTR_TRIGGER_TYPE attribute. Future versions of this specification cannot use these values:

101
102
103
104
105
106

Interpolation

Value Name	Language	Identifier	Actual Value
No Interpolation	C	IVISCOPE_VAL_NO_INTERPOLATION	1
	COM	IviScopeInterpolationNone	1
Sine X	C	IVISCOPE_VAL_SINE_X	2
	COM	IviScopeInterpolationSineX	2
Linear	C	IVISCOPE_VAL_LINEAR	3
	COM	IviScopeInterpolationLinear	3
Interpolation Class Ext Base	C	IVISCOPE_VAL_INTERPOLATION_ CLASS_EXT_BASE	100
Interpolation Specific Ext Base	C	IVISCOPE_VAL_INTERPOLATION_ SPECIFIC_EXT_BASE	1000
	COM	N/A	

TV Trigger Event

Value Name	Language	Identifier	Actual Value
TV Event Field1	C	IVISCOPE_VAL_TV_EVENT_FIELD1	1
	COM	IviScopeTVTriggerEventField1	1
TV Event Field2	C	IVISCOPE_VAL_TV_EVENT_FIELD2	2
	COM	IviScopeTVTriggerEventField2	2
TV Event Any Field	C	IVISCOPE_VAL_TV_EVENT_ANY_FIELD	3
	COM	IviScopeTVTriggerEventAnyField	3
TV Event Any Line	C	IVISCOPE_VAL_TV_EVENT_ANY_LINE	4
	COM	IviScopeTVTriggerEventAnyLine	4
TV Event Line Number	C	IVISCOPE_VAL_TV_EVENT_LINE_NUMBER	5
	COM	IviScopeTVTriggerEventLineNumber	5
TV Trigger Event Class Ext Base	C	IVISCOPE_VAL_TV_TRIGGER_EVENT_ CLASS_EXT_BASE	100
TV Trigger Event Specific Ext Base	C	IVISCOPE_VAL_TV_TRIGGER_EVENT_ SPECIFIC_EXT_BASE	1000
	COM	N/A	

The following values are reserved by the IviScope specification 1.0 for the IVISCOPE_ATTR_TV_TRIGGER_EVENT attribute. Future versions of this specification cannot use these values:

-1

TV Trigger Signal Format

Value Name	Language	Identifier	Actual Value
NTSC	C	IVISCOPE_VAL_NTSC	1
	COM	IviScopeTVSignalFormatNTSC	1
PAL	C	IVISCOPE_VAL_PAL	2
	COM	IviScopeTVSignalFormatPAL	2
SECAM	C	IVISCOPE_VAL_SECAM	3
	COM	IviScopeTVSignalFormatSECAM	3
TV Signal Format Class Ext Base	C	IVISCOPE_VAL_TV_SIGNAL_FORMAT_CLASS_EXT_BASE	100
TV Signal Format Specific Ext Base	C	IVISCOPE_VAL_TV_SIGNAL_FORMAT_SPECIFIC_EXT_BASE	1000
	COM	N/A	

TV Trigger Polarity

Value Name	Language	Identifier	Actual Value
TV Positive	C	IVISCOPE_VAL_TV_POSITIVE	1
	COM	IviScopeTVTriggerPolarityPositive	1
TV Negative	C	IVISCOPE_VAL_TV_NEGATIVE	2
	COM	IviScopeTVTriggerPolarityNegative	2
TV Trigger Polarity Class Ext Base	C	IVISCOPE_VAL_TV_TRIGGER_POLARITY_CLASS_EXT_BASE	100
TV Trigger Polarity Specific Ext Base	C	IVISCOPE_VAL_TV_TRIGGER_POLARITY_SPECIFIC_EXT_BASE	1000
	COM	N/A	

Runt Polarity

Value Name	Language	Identifier	Actual Value
Runt Positive	C	IVISCOPE_VAL_RUNT_POSITIVE	1
	COM	IviScopeRuntPolarityPositive	1
Runt Negative	C	IVISCOPE_VAL_RUNT_NEGATIVE	2
	COM	IviScopeRuntPolarityNegative	2
Runt Either	C	IVISCOPE_VAL_RUNT_EITHER	3
	COM	IviScopeRuntPolarityEither	3

Glitch Polarity

Value Name	Language	Identifier	Actual Value
Glitch Positive	C	IVISCOPE_VAL_GLITCH_POSITIVE	1
	COM	IviScopeGlitchPolarityPositive	1
Glitch Negative	C	IVISCOPE_VAL_GLITCH_NEGATIVE	2
	COM	IviScopeGlitchPolarityNegative	2
Glitch Either	C	IVISCOPE_VAL_GLITCH_EITHER	3
	COM	IviScopeGlitchPolarityEither	3

Glitch Condition

Value Name	Language	Identifier	Actual Value
Glitch Less Than	C	IVISCOPE_VAL_GLITCH_LESS_THAN	1
	COM	IviScopeGlitchConditionLessThan	1
Glitch Greater Than	C	IVISCOPE_VAL_GLITCH_GREATER_THAN	2
	COM	IviScopeGlitchConditionGreaterThan	2

Width Condition

Value Name	Language	Identifier	Actual Value
Width Within	C	IVISCOPE_VAL_WIDTH_WITHIN	1
	COM	IviScopeWidthConditionWithin	1
Width Outside	C	IVISCOPE_VAL_WIDTH_OUTSIDE	2
	COM	IviScopeWidthConditionOutside	2

Width Polarity

Value Name	Language	Identifier	Actual Value
Width Positive	C	IVISCOPE_VAL_WIDTH_POSITIVE	1
	COM	IviScopeWidthPolarityPositive	1
Width Negative	C	IVISCOPE_VAL_WIDTH_NEGATIVE	2
	COM	IviScopeWidthPolarityNegative	2
Width Either	C	IVISCOPE_VAL_WIDTH_EITHER	3
	COM	IviScopeWidthPolarityEither	3

AC Line Trigger Slope

Value Name	Language	Identifier	Actual Value
AC Line Positive	C	IVISCOPE_VAL_AC_LINE_POSITIVE	1
	COM	IviScopeACLinePositive	1
AC Line Negative	C	IVISCOPE_VAL_AC_LINE_NEGATIVE	2
	COM	IviScopeACLineNegative	2
AC Line Either	C	IVISCOPE_VAL_AC_LINE_EITHER	3
	COM	IviScopeACLineEither	3

Sample Mode

Value Name	Language	Identifier	Actual Value
Real Time	C	IVISCOPE_VAL_REAL_TIME	0
	COM	IviScopeSampleModeRealTime	0
Equivalent Time	C	IVISCOPE_VAL_EQUIVALENT_TIME	1
	COM	IviScopeSampleModeEquivalentTime	1

Trigger Modifier

Value Name	Language	Identifier	Actual Value
No Trigger Modifier	C	IVISCOPE_VAL_NO_TRIGGER_MOD	1
	COM	IviScopeTriggerModifierNone	1
Auto	C	IVISCOPE_VAL_AUTO	2
	COM	IviScopeTriggerModifierAuto	2
Auto Level	C	IVISCOPE_VAL_AUTO_LEVEL	3
	COM	IviScopeTriggerModifierAutoLevel	3
Trigger Modifier Class Ext Base	C	IVISCOPE_VAL_TRIGGER_MOD_ CLASS_EXT_BASE	100

Trigger Modifier Specific Ext Base	C	IVISCOPE_VAL_TRIGGER_MOD_ SPECIFIC_EXT_BASE	1000
	COM	N/A	

20.1 IviScope Obsolete Attribute Value Names

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

IVISCOPE_VAL_EDGE

IVISCOPE_VAL_WIDTH

IVISCOPE_VAL_RUNT

IVISCOPE_VAL_GLITCH

IVISCOPE_VAL_STATE

IVISCOPE_VAL_PATTERN

IVISCOPE_VAL_TV

IVISCOPE_VAL_IMMEDIATE

IVISCOPE_VAL_GPIB_GET

IVISCOPE_VAL_SW_TRIG_FUNC

IVISCOPE_VAL_AC_LINE

IVISCOPE_VAL_TV_SIGNAL_TYPE_CLASS_EXT_BASE

IVISCOPE_VAL_TV_SIGNAL_TYPE_SPECIFIC_EXT_BASE

IVISCOPE_VAL_TV_FIELD1

IVISCOPE_VAL_TV_FIELD2

IVISCOPE_VAL_TV_ANY_FIELD

IVISCOPE_VAL_TV_TRIGGER_FIELD_CLASS_EXT_BASE

IVISCOPE_VAL_TV_TRIGGER_FIELD_SPECIFIC_EXT_BASE

IVISCOPE_VAL_ACQ_TYPE_CLASS_EXT_BASE

IVISCOPE_VAL_ACQ_TYPE_SPECIFIC_EXT_BASE

IVISCOPE_VAL_INFINITE

IVISCOPE_VAL_50_OHMS

IVISCOPE_VAL_75_OHMS

IVISCOPE_VAL_1_MEG_OHM

21. IviScope Function Parameter Value Definitions

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

Read Waveform Measurement

Parameter: `Measurement`

Value Name	Language	Identifier	Actual Value
Rise Time	C	IVISCOPE_VAL_RISE_TIME	0
	COM	IviScopeMeasurementRiseTime	0
Fall Time	C	IVISCOPE_VAL_FALL_TIME	1
	COM	IviScopeMeasurementFallTime	1
Frequency	C	IVISCOPE_VAL_FREQUENCY	2
	COM	IviScopeMeasurementFrequency	2
Period	C	IVISCOPE_VAL_PERIOD	3
	COM	IviScopeMeasurementPeriod	3
Voltage Rms	C	IVISCOPE_VAL_VOLTAGE_RMS	4
	COM	IviScopeMeasurementVoltageRMS	4
Voltage Peak To Peak	C	IVISCOPE_VAL_VOLTAGE_PEAK_TO_PEAK	5
	COM	IviScopeMeasurementVoltagePeakToPeak	5
Voltage Max	C	IVISCOPE_VAL_VOLTAGE_MAX	6
	COM	IviScopeMeasurementVoltageMax	6
Voltage Min	C	IVISCOPE_VAL_VOLTAGE_MIN	7
	COM	IviScopeMeasurementVoltageMin	7
Voltage High	C	IVISCOPE_VAL_VOLTAGE_HIGH	8
	COM	IviScopeMeasurementVoltageHigh	8
Voltage Low	C	IVISCOPE_VAL_VOLTAGE_LOW	9
	COM	IviScopeMeasurementVoltageLow	9
Voltage Average	C	IVISCOPE_VAL_VOLTAGE_AVERAGE	10
	COM	IviScopeMeasurementVoltageAverage	10
Width Neg	C	IVISCOPE_VAL_WIDTH_NEG	11
	COM	IviScopeMeasurementWidthNeg	11
Width Pos	C	IVISCOPE_VAL_WIDTH_POS	12
	COM	IviScopeMeasurementWidthPos	12
Duty Cycle Neg	C	IVISCOPE_VAL_DUTY_CYCLE_NEG	13
	COM	IviScopeMeasurementDutyCycleNeg	13
Duty Cycle Pos	C	IVISCOPE_VAL_DUTY_CYCLE_POS	14
	COM	IviScopeMeasurementDutyCyclePos	14
Amplitude	C	IVISCOPE_VAL_AMPLITUDE	15
	COM	IviScopeMeasurementAmplitude	15

Voltage Cycle Rms	C	IVISCOPE_VAL_VOLTAGE_CYCLE_RMS	16
	COM	IviScopeMeasurementVoltageCycleRMS	16
Voltage Cycle Average	C	IVISCOPE_VAL_VOLTAGE_CYCLE_AVERAGE	17
	COM	IviScopeMeasurementVoltageCycleAverage	17
Overshoot	C	IVISCOPE_VAL_OVERSHOOT	18
	COM	IviScopeMeasurementOvershoot	18
Preshoot	C	IVISCOPE_VAL_PRESHOOT	19
	COM	IviScopeMeasurementPreshoot	19
Measurement Function Class Ext Base	C	IVISCOPE_VAL_MEASUREMENT_FUNCTION_CLASS_EXT_BASE	100
	COM	N/A	
Measurement Function Specific Ext Base	C	IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE	1000
	COM	N/A	

Parameter: MaxTimeMilliseconds

Value Name	Language	Identifier	Actual Value
Max Time Immediate	C	IVISCOPE_VAL_MAX_TIME_IMMEDIATE	0x0
	COM	IviScopeTimeOutImmediate	0x0
Max Time Infinite	C	IVISCOPE_VAL_MAX_TIME_INFINITE	0xFFFFFFFFUL
	COM	IviScopeTimeOutInfinite	0xFFFFFFFFUL

Fetch Waveform Measurement

Parameter: Measurement

The same as defined for the Measurement parameter of the Read Waveform Measurement function.

Read Waveform

Parameter: MaxTimeMilliseconds

The same as defined for the MaxTimeMilliseconds parameter of the Read Waveform Measurement function.

Read Min Max Waveform

Parameter: MaxTimeMilliseconds

The same as defined for the MaxTimeMilliseconds parameter of the Read Waveform Measurement function.

AcquisitionStatus

Parameter: `Status`

Value Name	Language	Identifier	Actual Value
Acquisition Complete	C	IVISCOPE_VAL_ACQ_COMPLETE	1
	COM	IviScopeAcquisitionStatusComplete	1
Acquisition In Progress	C	IVISCOPE_VAL_ACQ_IN_PROGRESS	0
	COM	IviScopeAcquisitionStatusInProgress	0
Acquisition Status Unknown	C	IVISCOPE_VAL_ACQ_STATUS_UNKNOWN	-1
	COM	IviScopeAcquisitionStatusUnknown	-1

21.1 IviScope Obsolete Function Parameter Value Names

The following attribute value names are reserved for backwards compatibility. Future versions of this specification cannot use these names:

IVISCOPE_VAL_CALCULATION_SPECIFIC_DRIVER_BASE

22. IviScope Error and Completion Code Value Definitions

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

Table 22-1. IviScope Completion Codes

Error Name	Description			
		Language	Identifier	Value (hex)
Invalid Waveform Element	One of the elements in the waveform array is invalid.			
		C	IVISCOPE_WARN_INVALID_WFM_ELEMENT	0x3FFA2001
		COM	S_IVISCOPE_INVALID_WFM_ELEMENT	0x00042001
Channel Not Enabled	Specified channel is not enabled.			
		C	IVISCOPE_ERROR_CHANNEL_NOT_ENABLED	0xBFFA2001
		COM	E_IVISCOPE_CHANNEL_NOT_ENABLED	0x80042001
Unable To Perform Measurement	Unable to perform desired measurement operation.			
		C	IVISCOPE_ERROR_UNABLE_TO_PERFORM_MEASUREMENT	0xBFFA2002
		COM	E_IVISCOPE_UNABLE_TO_PERFORM_MEASUREMENT	0x80042002
Max Time Exceeded	Maximum time exceeded before the operation completed.			
		C	IVISCOPE_ERROR_MAX_TIME_EXCEEDED	0xBFFA2003
		COM	E_IVISCOPE_MAX_TIME_EXCEEDED	0x80042003
Invalid Acquisition Type	Invalid acquisition type.			
		C	IVISCOPE_ERROR_INVALID_ACQ_TYPE	0xBFFA2004
		COM	E_IVISCOPE_INVALID_ACQ_TYPE	0x80042004

Table 22-2. IviScope Error Message Strings 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 22-2. IviScope Error Message Strings

Name	Message String
Invalid Waveform Element	"%s: Invalid waveform element
Channel Not Enabled	"%s: Channel not enabled
Unable To Perform Measurement	"%s: Unable to perform measurement
Max Time Exceeded	"%s: Maximum time exceeded"
Invalid Acquisition Type	"%s: Invalid acquisition type

23. IviScope Hierarchies

23.1 IviScope COM Hierarchy

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

Table 23-1. IviScope COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
Acquisition		
NumberOfAverages	Number of Averages	P
NumberOfEnvelopes	Number of Envelopes	P
Interpolation	Interpolation	P
Type	Type	P
ConfigureRecord	Configure Record	M
NumberOfPointsMin	Minimum Number of Points	P
StartTime	Start Time	P
TimePerRecord	Time Per Record	P
RecordLength	Record Length	P
SampleMode	Sample Mode	P
SampleRate	Sample Rate	P
Channels		
Count	Channel Count	P
Name	Channel Name	P
Item		
Configure	Configure Channel	M
ConfigureCharacteristics	Configure Channel Characteristics	M
Coupling	Channel Coupling	P
Enabled	Channel Enabled	P
Offset	Vertical Offset	P
Range	Vertical Range	P
ProbeAttenuation	Probe Attenuation	P
ProbeSense	Auto Probe Sense	P
InputFrequencyMax	Maximum Input Frequency	P
InputImpedance	Input Impedance	P
Measurements		
Count	Count	P
Name	Name	P
Abort	Abort Acquisition	M
Initiate	Initiate Acquisition	M
Status	Acquisition Status	M

Table 23-1. IviScope COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
IsWaveformElementInvalid	Is Invalid Waveform Element	M
AutoSetup	Auto Setup	M
Item		
FetchWaveform	Fetch Waveform	M
ReadWaveform	Read Waveform	M
FetchWaveformMinMax	Fetch Min Max Waveform	M
ReadWaveformMinMax	Read Min Max Waveform	M
FetchWaveformMeasurement	Fetch Waveform Measurement	M
ReadWaveformMeasurement	Read Waveform Measurement	M
ReferenceLevel		
Configure	Configure Reference Levels	M
Low	Low Reference Level	P
High	High Reference Level	P
Mid	Mid Reference Level	P
Trigger		
Continuous	Continuous Acquisition	P
Coupling	Trigger Coupling	P
Level	Trigger Level	P
Modifier	Trigger Modifier	P
Source	Trigger Source	P
Configure	Configure Trigger	M
Holdoff	Trigger Holdoff	P
Type	Trigger Type	P
ACLine		
Slope	AC Line Trigger Slope	P
Edge		
Configure	Configure Edge Trigger Source	M
Slope	Trigger Slope	P
Glitch		
Configure	Configure Glitch Trigger Source	M
Condition	Glitch Condition	P
Polarity	Glitch Polarity	P
Width	Glitch Width	P
Runt		
Configure	Configure Runt Trigger Source	M
Polarity	Runt Polarity	P
ThresholdLow	Runt Low Threshold	P
ThresholdHigh	Runt High Threshold	P

Table 23-1. IviScope COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
TV		
Configure	Configure TV Trigger Source	M
LineNumber	TV Trigger Line Number	P
Event	TV Trigger Event	P
Polarity	TV Trigger Polarity	P
SignalFormat	TV Trigger Signal Format	P
Width		
Configure	Configure Width Trigger Source	M
Condition	Width Condition	P
Polarity	Width Polarity	P
ThresholdLow	Width Low Threshold	P
ThresholdHigh	Width High Threshold	P

23.1.1 IviScope COM Interfaces

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

- IIviScopeAcquisition
- IIviScopeChannels
- IIviScopeMeasurements
- IIviScopeReferenceLevel
- IIviScopeTrigger

The IIviScopeTrigger interface contains interface reference properties for accessing additional the following IviScope trigger interfaces:

- IIviScopeTriggerACLine
- IIviScopeTriggerEdge
- IIviScopeTriggerGlitch
- IIviScopeTriggerRunt
- IIviScopeTriggerTV
- IIviScopeTriggerWidth

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

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

Table 23-2. IviScope Interface GUIDs lists the interfaces that this specification defines and their GUIDs.

Table 23-2. IviScope Interface GUIDs

Interface	GUID
IIviScope	{47ed524c-a398-11d4-ba58-000064657374}
IIviScopeAcquisition	{47ed524d-a398-11d4-ba58-000064657374}
IIviScopeChannels	{47ed524e-a398-11d4-ba58-000064657374}
IIviScopeChannel	{47ed524f-a398-11d4-ba58-000064657374}
IIviScopeMeasurements	{47ed5251-a398-11d4-ba58-000064657374}
IIviScopeMeasurement	{47ed5252-a398-11d4-ba58-000064657374}
IIviScopeReferenceLevel	{47ed5250-a398-11d4-ba58-000064657374}
IIviScopeTrigger	{47ed5253-a398-11d4-ba58-000064657374}
IIviScopeTriggerACLine	{47ed5254-a398-11d4-ba58-000064657374}
IIviScopeTriggerEdge	{47ed5255-a398-11d4-ba58-000064657374}
IIviScopeTriggerGlitch	{47ed5256-a398-11d4-ba58-000064657374}
IIviScopeTriggerRunt	{47ed5257-a398-11d4-ba58-000064657374}
IIviScopeTriggerTV	{47ed5258-a398-11d4-ba58-000064657374}
IIviScopeTriggerWidth	{47ed5259-a398-11d4-ba58-000064657374}

23.1.2 IviScope COM Interface Reference Properties

Interface reference properties are used to navigate the IviScope COM hierarchy. This section describes the interface reference properties that the IIviScope and IIviScopeTrigger interfaces define.

23.1.2.1 Acquisition

Data Type	Access
IIviScopeAcquisition*	RO

COM Property Name

Acquisition

Description

Returns a pointer to the IIviScopeAcquisition interface.

23.1.2.2 Channels

Data Type	Access
IIviScopeChannels*	RO

COM Property Name

Channels

Description

Returns a pointer to the IIviScopeChannels interface.

23.1.2.3 Measurements

Data Type	Access
IIviScopeMeasurements*	RO

COM Property Name

Measurements

Description

Returns a pointer to the IIviScopeMeasurements interface.

23.1.2.4 Reference Level

Data Type	Access
IIviScopeReferenceLevel*	RO

COM Property Name

ReferenceLevel

Description

Returns a pointer to the IIviScopeReferenceLevel interface.

23.1.2.5 Trigger

Data Type	Access
IIviScopeTrigger*	RO

COM Property Name

Trigger

Description

Returns a pointer to the IIviScopeTrigger interface.

23.1.2.6 AC Line Trigger

Data Type	Access
IIviScopeTriggerACLine*	RO

COM Property Name

Trigger.ACLine

Description

Returns a pointer to the IIviScopeTriggerACLine interface.

23.1.2.7 Edge Trigger

Data Type	Access
IIviScopeTriggerEdge*	RO

COM Property Name

Trigger.Edge

Description

Returns a pointer to the IIviScopeTriggerEdge interface.

23.1.2.8 Glitch Trigger

Data Type	Access
IIviScopeTriggerGlitch*	RO

COM Property Name

Trigger.Glitch

Description

Returns a pointer to the IIviScopeTriggerGlitch interface.

23.1.2.9 Runt Trigger

Data Type	Access
IIviScopeTriggerRunt*	RO

COM Property Name

Trigger.Runt

Description

Returns a pointer to the IIviScopeTriggerRunt interface.

23.1.2.10 TV

Data Type	Access
IIviScopeTriggerTV*	RO

COM Property Name

Trigger.TV

Description

Returns a pointer to the IIviScopeTriggerTV interface.

23.1.2.11 Width Trigger

Data Type	Access
IIviScopeTriggerWidth*	RO

COM Property Name

Trigger.Width

Description

Returns a pointer to the IIviScopeTriggerWidth interface.

23.1.3 IviScope COM Category

The IviScope class COM Category shall be “IviScope”, and the Category ID (CATID) shall be {47ed5156-a398-11d4-ba58-000064657374}.

23.2 IviScope C Function Hierarchy

The IviScope class function hierarchy is shown in the following table.

Name or Class	Function Name
Configuration…	
Acquisition…	
Configure Acquisition Type	IviScope_ConfigureAcquisitionType
Configure Acquisition Record	IviScope_ConfigureAcquisitionRecord
Configure Number of Averages	IviScope_ConfigureNumAverages
Configure Number of Envelopes	IviScope_ConfigureNumEnvelopes
Configure Interpolation	IviScope_ConfigureInterpolation
Configure Initiate Continuous	IviScope_ConfigureInitiateContinuous
Channel…	
Get Channel Name	IviScope_GetChannelName
Configure Channel	IviScope_ConfigureChannel
Configure Channel Characteristics	IviScope_ConfigureChanCharacteristics
Trigger…	
Configure Trigger	IviScope_ConfigureTrigger
Configure Trigger Coupling	IviScope_ConfigureTriggerCoupling
Configure Trigger Modifier	IviScope_ConfigureTriggerModifier
Configure Edge Trigger Source	IviScope_ConfigureEdgeTriggerSource
Configure TV Trigger Source	IviScope_ConfigureTVTriggerSource
Configure TV Trigger Ln Number	IviScope_ConfigureTVTriggerLineNumber
Configure Runt Trigger Source	IviScope_ConfigureRuntTriggerSource
Configure Glitch Trigger Source	IviScope_ConfigureGlitchTriggerSource
Configure Width Trigger Source	IviScope_ConfigureWidthTriggerSource
Configure AC Line Trigger Slope	IviScope_ConfigureAcLineTriggerSlope
Measurement	
Configure Reference Levels	IviScope_ConfigureRefLevels
Configuration Information…	
Actual Record Length	IviScope_ActualRecordLength
Auto Probe Sense Value	IviScope_AutoProbeSenseValue
Actual Sample Mode	IviScope_SampleMode
Actual Sample Rate	IviScope_SampleRate
Auto Setup	IviScope_AutoSetup
Waveform Acquisition…	

Name or Class	Function Name
Read Waveform	IviScope_ReadWaveform
Read Min/Max Waveform	IviScope_ReadMinMaxWaveform
Read Waveform Measurement	IviScope_ReadWaveformMeasurement
Low-Level Acquisition…	
Initiate Acquisition	IviScope_InitiateAcquisition
Acquisition Status	IviScope_AcquisitionStatus
Fetch Waveform	IviScope_FetchWaveform
Fetch Min/Max Waveform	IviScope_FetchMinMaxWaveform
Fetch Waveform Measurement	IviScope_FetchWaveformMeasurement
Abort	IviScope_Abort
Utility…	
Is Invalid Wfm Element	IviScope_IsInvalidWfmElement

23.2.1 IviScope Obsolete Function Names

The following function names are reserved by the IviScope specification 1.0. The future versions of this specification cannot use these names:

- `IviScope_ConfigureAcquisition`
- `IviScope_ConfigureEdgeTrigger`
- `IviScope_ConfigureGlitchTrigger`
- `IviScope_ConfigureHorizontal`
- `IviScope_ConfigureRuntTrigger`
- `IviScope_ConfigureTVTrigger`
- `IviScope_ConfigureTriggerSource`
- `IviScope_ConfigureVertical`
- `IviScope_ConfigureWidthTrigger`
- `IviScope_SendSWTrigger`

23.3 IviScope C Attribute Hierarchy

The IviScope class attribute hierarchy is shown in the following table.

Table 23-3. IviScope C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
Acquisition	
Acquisition Start Time	IVISCOPE_ATTR_ACQUISITION_START_TIME
Acquisition Type	IVISCOPE_ATTR_ACQUISITION_TYPE
Horizontal Minimum Number of Points	IVISCOPE_ATTR_HORZ_MIN_NUM_PTS
Horizontal Record Length	IVISCOPE_ATTR_HORZ_RECORD_LENGTH
Horizontal Sample Rate	IVISCOPE_ATTR_HORZ_SAMPLE_RATE
Horizontal Time Per Record	IVISCOPE_ATTR_HORZ_TIME_PER_RECORD
Interpolation	IVISCOPE_ATTR_INTERPOLATION
Sample Mode	IVISCOPE_ATTR_SAMPLE_MODE
Number of Averages	IVISCOPE_ATTR_NUM_AVERAGES
Number of Envelopes	IVISCOPE_ATTR_NUM_ENVELOPES
Initiate Continuous	IVISCOPE_ATTR_INITIATE_CONTINUOUS
Channel	
Channel Count	IVISCOPE_ATTR_CHANNEL_COUNT
Channel Enabled	IVISCOPE_ATTR_CHANNEL_ENABLED
Probe Attenuation	IVISCOPE_ATTR_PROBE_ATTENUATION
Probe Sense Value	IVISCOPE_ATTR_PROBE_SENSE_VALUE
Vertical Range	IVISCOPE_ATTR_VERTICAL_RANGE
Vertical Offset	IVISCOPE_ATTR_VERTICAL_OFFSET
Vertical Coupling	IVISCOPE_ATTR_VERTICAL_COUPLING
Maximum Input Frequency	IVISCOPE_ATTR_MAX_INPUT_FREQUENCY
Input Impedance	IVISCOPE_ATTR_INPUT_IMPEDANCE
Trigger	
Trigger Type	IVISCOPE_ATTR_TRIGGER_TYPE
Trigger Source	IVISCOPE_ATTR_TRIGGER_SOURCE
Trigger Coupling	IVISCOPE_ATTR_TRIGGER_COUPLING
Trigger Holdoff	IVISCOPE_ATTR_TRIGGER_HOLDOFF
Trigger Level	IVISCOPE_ATTR_TRIGGER_LEVEL
Trigger Modifier	IVISCOPE_ATTR_TRIGGER_MODIFIER
Edge Triggering	
Trigger Slope	IVISCOPE_ATTR_TRIGGER_SLOPE
TV Triggering	

Table 23-3. IviScope C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
TV Trigger Signal Format	IVISCOPE_ATTR_TV_TRIGGER_SIGNAL_FORMAT
TV Trigger Event	IVISCOPE_ATTR_TV_TRIGGER_EVENT
TV Trigger Line Number	IVISCOPE_ATTR_TV_TRIGGER_LINE_NUMBER
TV Trigger Polarity	IVISCOPE_ATTR_TV_TRIGGER_POLARITY
Runt Triggering	
Runt High Threshold	IVISCOPE_ATTR_RUNT_HIGH_THRESHOLD
Runt Low Threshold	IVISCOPE_ATTR_RUNT_LOW_THRESHOLD
Runt Polarity	IVISCOPE_ATTR_RUNT_POLARITY
Glitch Triggering	
Glitch Width	IVISCOPE_ATTR_GLITCH_WIDTH
Glitch Polarity	IVISCOPE_ATTR_GLITCH_POLARITY
Glitch Condition	IVISCOPE_ATTR_GLITCH_CONDITION
Width Triggering	
Width Condition	IVISCOPE_ATTR_WIDTH_CONDITION
Width High Threshold	IVISCOPE_ATTR_WIDTH_HIGH_THRESHOLD
Width Low Threshold	IVISCOPE_ATTR_WIDTH_LOW_THRESHOLD
Width Polarity	IVISCOPE_ATTR_WIDTH_POLARITY
AC Line Triggering	
AC Line Trigger Slope	IVISCOPE_ATTR_AC_LINE_TRIGGER_SLOPE
Waveform Measurement	
Measurement High Reference	IVISCOPE_ATTR_MEAS_HIGH_REF
Measurement Low Reference	IVISCOPE_ATTR_MEAS_LOW_REF
Measurement Middle Reference	IVISCOPE_ATTR_MEAS_MID_REF

Appendix A. Specific Driver 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 IviScope class.

A.2 Disabling Unused Extensions

IVI Class-Compliant specific drivers are required to disable extension capability groups that an application program does not explicitly use. The IVI Class-Compliant specific driver can do so by setting the attributes of an extension capability group to the values that this section recommends. An IVI Class-Compliant specific driver can set these values for all extension capability groups when the Initialize, and 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, an IVI Class-Compliant 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 IVI Class-Compliant 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 IviScopeInterpolation Extension Group

Attribute value that effectively disables the IviScopeInterpolation extension group is shown in the following table.

Table 23-4. Values for Disabling the IviScopeInterpolation Extension Group

Attribute	Value
Interpolation	No Interpolation

Disabling the IviScopeTVTrigger Extension Group

The IviScopeTVTrigger extension group affects the instrument behavior only when the Trigger Type attribute is set to TV Trigger value. Therefore, this specification does not recommend attribute values that disable the IviScopeTVTrigger extension group.

Disabling the IviScopeRuntTrigger Extension Group

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

Disabling the IviScopeGlitchTrigger Extension Group

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

Disabling the IviScopeWidthTrigger Extension Group

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

Disabling the IviScopeACLineTrigger Extension Group

The IviScopeACLineTrigger extension group affects the instrument behavior only when the Trigger Type attribute is set to AC Line Trigger. Therefore, this specification does not recommend attribute values that disable the IviScopeACLineTrigger extension group.

Disabling the IviScopeWaveformMeasurement Extension Group

The IviScopeWaveformMeasurement extension group affects the instrument behavior only when the end user calls the Read Waveform Measurement or Fetch Waveform Measurement functions. Therefore, this specification does not recommend attribute values that disable the IviScopeWaveformMeasurement extension group.

Disabling the IviScopeMinMaxWaveform Extension Group

The IviScopeMinMaxWaveform extension group affects the instrument behavior only when the Acquisition Type attribute is set to Envelope or Peak Detect. Therefore, this specification does not recommend attribute values that disable the IviScopeMinMaxWaveform extension group.

Disabling the IviScopeProbeAutoSense Extension Group

The IviScopeProbeAutoSense extension group defines the read-only attributes and functions that analyze the instrument's state. These attributes and functions do not affect the instrument behavior.

Disabling the IviScopeContinuousAcquisition Extension Group

Attribute value that effectively disables the IviScopeContinuousAcquisition extension group is shown in the following table.

Table 23-5. Values for Disabling the IviScopeContinuousAcquisition Extension Group

Attribute	Value
Initiate Continuous	False

Disabling the IviScopeAverageAcquisition Extension Group

The IviScopeAverageAcquisition extension group affects the instrument behavior only when the Acquisition Type attribute is set to Average. Therefore, this specification does not recommend attribute values that disable the IviScopeAverageAcquisition extension group.

Disabling the IviScopeSampleMode Extension Group

The IviScopeSampleMode extension group defines the read-only attributes and functions that analyze the instrument's state. These attributes and functions do not affect the instrument behavior.

Disabling the IviScopeTriggerModifier Extension Group

Attribute value that effectively disables the IviScopeTriggerModifier extension group is shown in the following table.

Table 23-6. Values for Disabling the IviScopeTriggerModifier Extension Group

Attribute	Value
Trigger Modifier	No Trigger Mod

Disabling the IviScopeAutoSetup Extension Group

The IviScopeAutoSetup extension group affects the instrument behavior only when the end user calls the Auto Setup function. Therefore, this specification does not recommend attribute values that disable the IviScopeAutoSetup extension group.

A.3 Query Instrument Status

Based on the value of Query Instrument Status, the IVI Class-Compliant specific driver may check the status of the instrument to see if it has encountered an error. In IVI Class-Compliant specific driver functions, the status check should not occur in the lowest-level signal generation functions Initiate Acquisition, Abort, Fetch Waveform, Fetch Min Max Waveform, and Fetch Waveform Measurement. These functions are intended to give the application developer low-level control over signal generation. When calling these functions, the application developer is responsible for checking the status of the instrument. Checking status in every function at this level would also add unnecessary overhead to the specific instrument driver.

A.4 Relationship of Acquisition Type and Horizontal Minimum Number of Points attributes

The end-user sets the Horizontal Minimum Number of Points attribute to specify the number of points they want to acquire into the oscilloscope's waveform record. Most oscilloscopes accept only a discrete set of record lengths, but not necessarily the same discrete set. The set of acceptable record lengths varies from one oscilloscope to another. You must make sure that the value the driver writes to the instrument is the exact number that the instrument accepts, regardless of whether the oscilloscope coerces values. This ensures that the cache value of the attribute truly reflects the state of the oscilloscope. At the same time, the driver must remember the minimum number of points the user requests. It is possible for other settings to change the set of valid record lengths. When the user initiates an acquisition, the driver must verify that the current record length is greater than or equal to the minimum number of points.

A.5 Auto-Setup and attribute invalidations

When the end-user calls the Auto Setup function, the oscilloscope senses the input signal and configures many of the instrument settings automatically. If the driver is using state caching, the cache value of the instrument driver attributes that correspond to the settings the oscilloscope changes no longer reflect the state of the instrument. Therefore, you must invalidate the cache value of all the attributes. You can do this with the Invalidate All Attributes function.

A.6 Suggestions for Implementing the Probe Attenuation Attribute

You must pay particular attention to oscilloscopes that have the ability to sense the attenuation of the probe and to adjust other settings accordingly. The possible approach is the implementation of two additional attributes, Probe Sense Value and Probe Sense in your driver. Probe Sense Value is an extension attribute defined in the IviScopeProbeAutoSense Extension section of this specification. Probe Sense is an instrument specific attribute that is used internally by the driver to record whether the instrument is currently using the probe sensing capability. The driver uses these attributes to calculate the valid ranges for other attributes that are dependent upon the probe attenuation attribute. These attributes include Vertical Range, Vertical Offset, and all the attributes that configure trigger levels.

The different scenarios include:

- The oscilloscope does not have the probe sense capability. The only valid settings for the Probe

Attenuation attribute are manual probe attenuation values.

To handle this case, do not allow the Probe Attenuation attribute to be set to the Probe Sense On value.

- The oscilloscope can sense the probe attenuation, and it also allows the user to specify a manual attenuation setting.

  To handle this case, implement the Probe Sense Value and Probe Sense attributes. When the end-user enables the automatic probe sense capability or the instrument driver reads form the instrument that automatic probe sense is enabled, and the driver implements the state caching, the driver must disable state caching for the dependent attributes. When the end-user disables the automatic probe sense capability or the instrument driver reads from the instrument that it is using manual probe attenuation, the driver must enable state caching for the dependent attributes.

- The oscilloscope can sense the probe attenuation, but it does not accept any manual settings.

  Allow the users to specify manual settings and assume that the instrument operator does not replace the probe after the application program specifies the manual setting. Notice that this is the same assumption that you must always make when using manual settings on any oscilloscope.

  To allow for manual settings, check if the user sets the attribute to an acceptable value. The acceptable values are Probe Sense On or the actual probe attenuation the oscilloscope is currently sensing. You must implement the Probe Sense Value and `IVISCOPE_ATTR_PROBE_SENSE` attributes as in the previous case.

A.7 Attributes that use the Probe Attenuation attribute

Many attributes depend on the value of the Probe Attenuation attribute. These attributes include the Vertical Range, Vertical Offset, and Trigger Level attributes. Attributes that depend on the value of the Probe Attenuation attribute typically use the probe attenuation value to check and coerce the value the end-user specifies for them. When you implement these attributes, make sure that you always perform the checking against the actual probe attenuation value.

A.8 Relationship of the Vertical Coupling and Trigger Coupling attributes.

In most cases, oscilloscopes have the circuitry for performing the coupling on the acquisition channel separate from the circuitry that performs the coupling on the trigger subsystem. This specification assumes that the instrument can set the channel coupling independently from the trigger coupling. If this is not the case, the IVI Class-Compliant specific driver must implement the Vertical Coupling as the dominant attribute. This means that setting this attribute to a value allowed by the instrument always results in the successful operation. On such instruments, setting the Trigger Coupling attribute to anything that conflicts with the specified channel coupling must cause an Invalid Value error.

A.9 Instruments that have channel-based record lengths.

Most oscilloscopes have the same record length on all acquisition channels. This specification assumes that the setting for the record length is a global setting for all channels, and it uses Horizontal Minimum Number of Points and Horizontal Record Length to model this behavior. If the instrument has different acquisition lengths on different channels, then the class behavior would dictate setting all of the channels to the same value with the Horizontal Minimum Number of Points. The different record lengths must be modeled as an instrument specific behavior.

A.10 Implementing the Trigger Holdoff attribute

This specification defines the hold-off as the length of time the oscilloscope waits after it detects a trigger until it responds to additional triggers. Many analog oscilloscopes define the hold-off as starting from end of the previous waveform acquisition instead of from the previous trigger. Some digital oscilloscopes specify the hold-off from the end of the waveform acquisition as well.

These differences in how oscilloscopes specify the hold-off setting can lead to non-interchangeable instruments behavior. Therefore if your instrument defines the hold-off as starting from the end of the previous waveform acquisition, you **must** translate that hold-off time to the one defined in this specification. To do that, you may do the following:

1. Translate the value the end-user specifies for hold-off to a value your oscilloscope expects when you implement the attribute. You can do this by subtracting the length of time from the Trigger Event to the end of the waveform record from the hold-off value the user specifies. Then send the resulting number to the oscilloscope. If the number is less than 0.0, use 0.0.
2. Perform the opposite translation when obtaining the value from the instrument.
3. Since it now depends on the acquisition settings, make sure that setting the Acquisition Start Time, Horizontal Time Per Second, and Horizontal Minimum Number of Points attributes invalidates the Trigger Holdoff attribute.

Appendix B. Interchangeability Checking Rules

B.1 Introduction

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

B.2 When to Perform Interchangeability Checking

Refer to Section 3.3.6*: Interchangeability Checking in IVI-3.1: Driver Architecture Specification* for a description of the rules for interchangeability checking in IVI drivers. The remainder of this section defines additional rules and exceptions for each capability group.

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

- The Interchange Check attribute is set to True
- The user calls one of the following functions.
  - Initiate Acquisition
  - Read Waveform
  - Read Min Max Waveform
  - Read Waveform Measurement

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.

IviScopeBase Capability Group

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

IviScopeInterpolation Capability Group

The Interpolation attribute must be in a user specified state only if the application sets the Acquisition Type attribute to Average.

IviScopeTVTrigger Capability Group

1. The driver performs interchangeability checking on the IviScopeTVTrigger group only if the application sets the Trigger Type attribute to TV Trigger.

2. The TV Trigger Line Number attribute must be in a user specified state only if the application sets the TV Trigger Event attribute to TV Event Line Number.

IviScopeRuntTrigger Capability Group

1. The driver performs interchangeability checking on the IviScopeRuntTrigger group only if the application sets the Trigger Type attribute to Runt Trigger.
2. The Trigger Level attribute must be in a user-specified state only if the application sets the Trigger Type attribute to Runt Trigger.

IviScopeGlitchTrigger Capability Group

The driver performs interchangeability checking on the IviScopeGlitchTrigger group only if the application sets the Trigger Type attribute to Glitch Trigger.

IviScopeWidthTrigger Capability Group

The driver performs interchangeability checking on the IviScopeWidthTrigger group only if the application sets the Trigger Type attribute to Width Trigger.

IviScopeAcLineTrigger Capability Group

The driver performs interchangeability checking on the IviScopeAcLineTrigger group only if the application sets the Trigger Type attribute to AC Line Trigger.

IviScopeWaveformMeas Capability Group

1. The Measurement Low Reference attribute must be in a user-specified state if the user requests a waveform measurement that requires the low reference level, such as rise time, fall time, preshoot and overshoot.
2. The Measurement Middle Reference attribute must be in a user-specified state if the user requests a waveform measurement that requires the middle reference level, such as frequency, period, positive and negative pulse widths, and all cycle-based measurements.
3. The Measurement High Reference attribute must be in a user-specified state if the user requests a waveform measurement that requires the high reference level, such as rise time, fall time, preshoot and overshoot.

IviScopeMinMaxWaveform Capability Group

1. The driver performs interchangeability checking on the IviScopeMinMaxWaveform group only if the application sets the Acquisition Type attribute to Envelope or Peak Detect.
3. The Number of Envelopes attribute must be in a user-specified state only if the application sets the Acquisition Type attribute to Envelope.

IviScopeProbeAutoSense Capability Group

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

IviScopeContinuousAcquisition Capability Group

Using this extension group is inherently non-interchangeable. This specification does not define the behavior of the instrument nor the data the instruments return while continuously acquiring the data.

IviScopeAverageAcquisition Capability Group

The driver performs interchangeability checking on the IviScopeAverageAcquisition group only if the application sets the Acquisition Type attribute to Average.

IviScopeSampleMode Capability Group

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

IviScopeTriggerModifier Capability Group

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

IviScopeAutoSetup Capability Group

Using this extension group is inherently non-interchangeable. The application behavior then depends on the internal instrument's algorithms for optimal acquisition and channel configuration.

Appendix C. ANSI C Include File

The C source code below provides an example of how a class driver C interface might be defined. It provides definitions only for attributes, functions, values, and status codes that this specification defines. It does not represent a complete interface for an IviScope compliant driver. To aid in the creation of an IviScope-compliant specific driver, replace `IVISCOPE` with the actual driver prefix using uppercase characters and replace `IviScope` with consistent case sensitivity.

```
/*****************************************************************************
 *                              I V I - S C O P E                            *
 *                                                                           *
 * Title:       IviScope include file                                        *
 * Purpose:     IviScope Class declarations for Base and Extended Capabilities.
 *                                                                           *
 *                                                                           *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001        *
 * All rights reserved.                                                      *
 *****************************************************************************/


#ifndef IVISCOPE_HEADER
#define IVISCOPE_HEADER

#include <ivic.h>

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

/*****************************************************************************
 *---------------------- IviScope Class Attribute Defines ---------------------*
 *****************************************************************************/


    /*- IviScopeBase Attributes -*/
#define IVISCOPE_ATTR_CHANNEL_COUNT             (IVI_INHERENT_ATTR_BASE + 203)
#define IVISCOPE_ATTR_VERTICAL_RANGE            (IVI_CLASS_ATTR_BASE  + 1)
#define IVISCOPE_ATTR_VERTICAL_OFFSET           (IVI_CLASS_ATTR_BASE  + 2)
#define IVISCOPE_ATTR_VERTICAL_COUPLING         (IVI_CLASS_ATTR_BASE  + 3)
#define IVISCOPE_ATTR_PROBE_ATTENUATION         (IVI_CLASS_ATTR_BASE  + 4)
#define IVISCOPE_ATTR_CHANNEL_ENABLED           (IVI_CLASS_ATTR_BASE  + 5)
#define IVISCOPE_ATTR_MAX_INPUT_FREQUENCY       (IVI_CLASS_ATTR_BASE  + 6)
#define IVISCOPE_ATTR_INPUT_IMPEDANCE           (IVI_CLASS_ATTR_BASE  + 103)

        /*- Acquisition Subsystem -*/
#define IVISCOPE_ATTR_ACQUISITION_TYPE          (IVI_CLASS_ATTR_BASE  + 101)
#define IVISCOPE_ATTR_ACQUISITION_START_TIME    (IVI_CLASS_ATTR_BASE  + 109)
#define IVISCOPE_ATTR_HORZ_TIME_PER_RECORD      (IVI_CLASS_ATTR_BASE  + 7)
#define IVISCOPE_ATTR_HORZ_RECORD_LENGTH        (IVI_CLASS_ATTR_BASE  + 8)
#define IVISCOPE_ATTR_HORZ_MIN_NUM_PTS          (IVI_CLASS_ATTR_BASE  + 9)
#define IVISCOPE_ATTR_HORZ_SAMPLE_RATE          (IVI_CLASS_ATTR_BASE  + 10)

        /*- Triggering Subsystem -*/
#define IVISCOPE_ATTR_TRIGGER_TYPE              (IVI_CLASS_ATTR_BASE  + 12)
#define IVISCOPE_ATTR_TRIGGER_SOURCE            (IVI_CLASS_ATTR_BASE  + 13)
#define IVISCOPE_ATTR_TRIGGER_COUPLING          (IVI_CLASS_ATTR_BASE  + 14)
#define IVISCOPE_ATTR_TRIGGER_HOLDOFF           (IVI_CLASS_ATTR_BASE  + 16)

        /*- Edge Triggering Attributes -*/
#define IVISCOPE_ATTR_TRIGGER_LEVEL             (IVI_CLASS_ATTR_BASE  + 17)
#define IVISCOPE_ATTR_TRIGGER_SLOPE             (IVI_CLASS_ATTR_BASE  + 18)

    /*- IviScope Extended Attributes -*/
        /*- IviScopeTVTrigger Extension Group -*/
#define IVISCOPE_ATTR_TV_TRIGGER_SIGNAL_FORMAT  (IVI_CLASS_ATTR_BASE + 201)
#define IVISCOPE_ATTR_TV_TRIGGER_EVENT          (IVI_CLASS_ATTR_BASE + 205)
#define IVISCOPE_ATTR_TV_TRIGGER_LINE_NUMBER    (IVI_CLASS_ATTR_BASE + 206)
#define IVISCOPE_ATTR_TV_TRIGGER_POLARITY       (IVI_CLASS_ATTR_BASE + 204)
```

```
        /*- IviScopeRuntTrigger Extension Group -*/
#define IVISCOPE_ATTR_RUNT_HIGH_THRESHOLD       (IVI_CLASS_ATTR_BASE + 301)
#define IVISCOPE_ATTR_RUNT_LOW_THRESHOLD        (IVI_CLASS_ATTR_BASE + 302)
#define IVISCOPE_ATTR_RUNT_POLARITY             (IVI_CLASS_ATTR_BASE + 303)

        /*- IviScopeGlitchTrigger Extension Group -*/
#define IVISCOPE_ATTR_GLITCH_WIDTH              (IVI_CLASS_ATTR_BASE + 401)
#define IVISCOPE_ATTR_GLITCH_POLARITY           (IVI_CLASS_ATTR_BASE + 402)
#define IVISCOPE_ATTR_GLITCH_CONDITION          (IVI_CLASS_ATTR_BASE + 403)

        /*- IviScopeWidthTrigger Extension Group -*/
#define IVISCOPE_ATTR_WIDTH_LOW_THRESHOLD       (IVI_CLASS_ATTR_BASE + 501)
#define IVISCOPE_ATTR_WIDTH_HIGH_THRESHOLD      (IVI_CLASS_ATTR_BASE + 502)
#define IVISCOPE_ATTR_WIDTH_POLARITY            (IVI_CLASS_ATTR_BASE + 503)
#define IVISCOPE_ATTR_WIDTH_CONDITION           (IVI_CLASS_ATTR_BASE + 504)

        /*- IviScopeAcLineTrigger Extension Group -*/
#define IVISCOPE_ATTR_AC_LINE_TRIGGER_SLOPE     (IVI_CLASS_ATTR_BASE + 701)

        /*- IviScopeMinMaxWaveform Extension Group -*/
#define IVISCOPE_ATTR_NUM_ENVELOPES             (IVI_CLASS_ATTR_BASE + 105)

        /*- IviScopeWaveformMeas Extension Group -*/
#define IVISCOPE_ATTR_MEAS_HIGH_REF             (IVI_CLASS_ATTR_BASE + 607)
#define IVISCOPE_ATTR_MEAS_LOW_REF              (IVI_CLASS_ATTR_BASE + 608)
#define IVISCOPE_ATTR_MEAS_MID_REF              (IVI_CLASS_ATTR_BASE + 609)

        /*- IviScope Trigger Modifier Extension Group -*/
#define IVISCOPE_ATTR_TRIGGER_MODIFIER          (IVI_CLASS_ATTR_BASE + 102)

        /*- IviScope Average Acquisition Extension Group -*/
#define IVISCOPE_ATTR_NUM_AVERAGES              (IVI_CLASS_ATTR_BASE + 104)

        /*- IviScope Sample Mode Extension Group -*/
#define IVISCOPE_ATTR_SAMPLE_MODE               (IVI_CLASS_ATTR_BASE + 106)

        /*- IviScope Continuous Acquisition Extension Group -*/
#define IVISCOPE_ATTR_INITIATE_CONTINUOUS       (IVI_CLASS_ATTR_BASE + 107)

        /*- IviScope Probe Auto Sense Extension Group -*/
#define IVISCOPE_ATTR_PROBE_SENSE_VALUE         (IVI_CLASS_ATTR_BASE + 108)

        /*- IviScope Interpolation Extension Group -*/
#define IVISCOPE_ATTR_INTERPOLATION             (IVI_CLASS_ATTR_BASE  + 19)

/*****************************************************************************
 *------- IviScope Class Function Parameter and Attribute Value Defines ------*
 *****************************************************************************/

   /*- Defined values for MaxTimeMilliseconds parameter to the waveform acquisition and
measurement functions -*/
#define IVISCOPE_VAL_MAX_TIME_INFINITE                          (-1)
#define IVISCOPE_VAL_MAX_TIME_IMMEDIATE                         (0)

    /*- Defined values for the status parameter of the IviScope_AcquisitionStatus function -*/
#define IVISCOPE_VAL_ACQ_COMPLETE                               (1)
#define IVISCOPE_VAL_ACQ_IN_PROGRESS                            (0)
#define IVISCOPE_VAL_ACQ_STATUS_UNKNOWN                         (-1)

    /*- Defined values for the measurementFunction parameter of the
IviScope_ReadWaveformMeasurment function -*/
#define IVISCOPE_VAL_RISE_TIME                                  (0)
#define IVISCOPE_VAL_FALL_TIME                                  (1)
#define IVISCOPE_VAL_FREQUENCY                                  (2)
#define IVISCOPE_VAL_PERIOD                                     (3)
#define IVISCOPE_VAL_VOLTAGE_RMS                                (4)
#define IVISCOPE_VAL_VOLTAGE_PEAK_TO_PEAK                       (5)
#define IVISCOPE_VAL_VOLTAGE_MAX                                (6)
#define IVISCOPE_VAL_VOLTAGE_MIN                                (7)
#define IVISCOPE_VAL_VOLTAGE_HIGH                               (8)
```

```
#define IVISCOPE_VAL_VOLTAGE_LOW                                (9)
#define IVISCOPE_VAL_VOLTAGE_AVERAGE                            (10)
#define IVISCOPE_VAL_WIDTH_NEG                                  (11)
#define IVISCOPE_VAL_WIDTH_POS                                  (12)
#define IVISCOPE_VAL_DUTY_CYCLE_NEG                             (13)
#define IVISCOPE_VAL_DUTY_CYCLE_POS                             (14)
#define IVISCOPE_VAL_AMPLITUDE                                  (15)
#define IVISCOPE_VAL_VOLTAGE_CYCLE_RMS                          (16)
#define IVISCOPE_VAL_VOLTAGE_CYCLE_AVERAGE                      (17)
#define IVISCOPE_VAL_OVERSHOOT                                  (18)
#define IVISCOPE_VAL_PRESHOOT                                   (19)

#define IVISCOPE_VAL_MEASUREMENT_FUNCTION_CLASS_EXT_BASE        (100)
#define IVISCOPE_VAL_MEASUREMENT_FUNCTION_SPECIFIC_EXT_BASE     (1000)

    /*- Defined values for IVISCOPE_ATTR_VERTICAL_COUPLING -*/
#define IVISCOPE_VAL_AC                                         (0)
#define IVISCOPE_VAL_DC                                         (1)
#define IVISCOPE_VAL_GND                                        (2)
#define IVISCOPE_VAL_VERTICAL_COUPLING_CLASS_EXT_BASE           (100)
#define IVISCOPE_VAL_VERTICAL_COUPLING_SPECIFIC_EXT_BASE        (1000)

    /*- Defined values for IVISCOPE_ATTR_TRIGGER_TYPE -*/
#define IVISCOPE_VAL_EDGE_TRIGGER                               (1)
#define IVISCOPE_VAL_WIDTH_TRIGGER                              (2)
#define IVISCOPE_VAL_RUNT_TRIGGER                               (3)
#define IVISCOPE_VAL_GLITCH_TRIGGER                             (4)
#define IVISCOPE_VAL_TV_TRIGGER                                 (5)
#define IVISCOPE_VAL_IMMEDIATE_TRIGGER                          (6)
#define IVISCOPE_VAL_AC_LINE_TRIGGER                            (7)
#define IVISCOPE_VAL_TRIGGER_TYPE_CLASS_EXT_BASE                (200)
#define IVISCOPE_VAL_TRIGGER_TYPE_SPECIFIC_EXT_BASE             (1000)


    /*- Defined values for IVISCOPE_ATTR_TRIGGER_SLOPE -*/
#define IVISCOPE_VAL_POSITIVE                                   (1)
#define IVISCOPE_VAL_NEGATIVE                                   (0)

    /*- Defined values for IVISCOPE_ATTR_TRIGGER_SOURCE -*/
#define IVISCOPE_VAL_EXTERNAL                       "VAL_EXTERNAL"
#define IVISCOPE_VAL_TTL0                           "VAL_TTL0"
#define IVISCOPE_VAL_TTL1                           "VAL_TTL1"
#define IVISCOPE_VAL_TTL2                           "VAL_TTL2"
#define IVISCOPE_VAL_TTL3                           "VAL_TTL3"
#define IVISCOPE_VAL_TTL4                           "VAL_TTL4"
#define IVISCOPE_VAL_TTL5                           "VAL_TTL5"
#define IVISCOPE_VAL_TTL6                           "VAL_TTL6"
#define IVISCOPE_VAL_TTL7                           "VAL_TTL7"
#define IVISCOPE_VAL_ECL0                           "VAL_ECL0"
#define IVISCOPE_VAL_ECL1                           "VAL_ECL1"
#define IVISCOPE_VAL_PXI_STAR                       "VAL_PXI_STAR"
#define IVISCOPE_VAL_RTSI_0                         "VAL_RTSI_0"
#define IVISCOPE_VAL_RTSI_1                         "VAL_RTSI_1"
#define IVISCOPE_VAL_RTSI_2                         "VAL_RTSI_2"
#define IVISCOPE_VAL_RTSI_3                         "VAL_RTSI_3"
#define IVISCOPE_VAL_RTSI_4                         "VAL_RTSI_4"
#define IVISCOPE_VAL_RTSI_5                         "VAL_RTSI_5"
#define IVISCOPE_VAL_RTSI_6                         "VAL_RTSI_6"

    /*
        In addition to the above defines,
        IVISCOPE_ATTR_TRIGGER_SOURCE accepts any defined
        channel name or string representation of a channel number
    */

    /*- Defined extended values for IVISCOPE_ATTR_PROBE_ATTENUATION -*/
#define IVISCOPE_VAL_PROBE_SENSE_ON                             (-1)

#define IVISCOPE_VAL_PROBE_ATTENUATION_CLASS_EXT_BASE           (-100)
#define IVISCOPE_VAL_PROBE_ATTENUATION_SPECIFIC_EXT_BASE        (-1000)
```

```
    /*- Defined values for IVISCOPE_ATTR_TRIGGER_COUPLING -*/
/* #define IVISCOPE_VAL_AC                     DEFINED ABOVE */
/* #define IVISCOPE_VAL_DC                     DEFINED ABOVE */
#define IVISCOPE_VAL_HF_REJECT                              (3)
#define IVISCOPE_VAL_LF_REJECT                              (4)
#define IVISCOPE_VAL_NOISE_REJECT                           (5)

#define IVISCOPE_VAL_TRIGGER_COUPLING_CLASS_EXT_BASE        (100)
#define IVISCOPE_VAL_TRIGGER_COUPLING_SPECIFIC_EXT_BASE     (1000)

    /*- Defined values for IVISCOPE_ATTR_INTERPOLATION -*/
#define IVISCOPE_VAL_NO_INTERPOLATION                       (1)
#define IVISCOPE_VAL_SINE_X                                 (2)
#define IVISCOPE_VAL_LINEAR                                 (3)
#define IVISCOPE_VAL_INTERPOLATION_CLASS_EXT_BASE           (100)
#define IVISCOPE_VAL_INTERPOLATION_SPECIFIC_EXT_BASE        (1000)

    /*- Defined values for IVISCOPE_ATTR_TV_TRIGGER_SIGNAL_FORMAT -*/
#define IVISCOPE_VAL_NTSC                                   (1)
#define IVISCOPE_VAL_PAL                                    (2)
#define IVISCOPE_VAL_SECAM                                  (3)
#define IVISCOPE_VAL_TV_SIGNAL_FORMAT_CLASS_EXT_BASE        (100)
#define IVISCOPE_VAL_TV_SIGNAL_FORMAT_SPECIFIC_EXT_BASE     (1000)

    /*- Defined values for IVISCOPE_ATTR_TV_TRIGGER_EVENT -*/
#define IVISCOPE_VAL_TV_EVENT_FIELD1                        (1)
#define IVISCOPE_VAL_TV_EVENT_FIELD2                        (2)
#define IVISCOPE_VAL_TV_EVENT_ANY_FIELD                     (3)
#define IVISCOPE_VAL_TV_EVENT_ANY_LINE                      (4)
#define IVISCOPE_VAL_TV_EVENT_LINE_NUMBER                   (5)
#define IVISCOPE_VAL_TV_TRIGGER_EVENT_CLASS_EXT_BASE        (100)
#define IVISCOPE_VAL_TV_TRIGGER_EVENT_SPECIFIC_EXT_BASE     (1000)

    /*- Defined values for IVISCOPE_ATTR_TV_TRIGGER_POLARITY -*/
#define IVISCOPE_VAL_TV_POSITIVE                            (1)
#define IVISCOPE_VAL_TV_NEGATIVE                            (2)
#define IVISCOPE_VAL_TV_TRIGGER_POLARITY_CLASS_EXT_BASE     (100)
#define IVISCOPE_VAL_TV_TRIGGER_POLARITY_SPECIFIC_EXT_BASE  (1000)

    /*- Defined values for IVISCOPE_ATTR_RUNT_POLARITY -*/
#define IVISCOPE_VAL_RUNT_POSITIVE                          (1)
#define IVISCOPE_VAL_RUNT_NEGATIVE                          (2)
#define IVISCOPE_VAL_RUNT_EITHER                            (3)

    /*- Defined values for IVISCOPE_ATTR_GLITCH_POLARITY -*/
#define IVISCOPE_VAL_GLITCH_POSITIVE                         (1)
#define IVISCOPE_VAL_GLITCH_NEGATIVE                         (2)
#define IVISCOPE_VAL_GLITCH_EITHER                           (3)

    /*- Defined values for IVISCOPE_ATTR_GLITCH_CONDITION -*/
#define IVISCOPE_VAL_GLITCH_LESS_THAN                        (1)
#define IVISCOPE_VAL_GLITCH_GREATER_THAN                     (2)

    /*- Defined values for IVISCOPE_ATTR_WIDTH_POLARITY -*/
#define IVISCOPE_VAL_WIDTH_POSITIVE                          (1)
#define IVISCOPE_VAL_WIDTH_NEGATIVE                          (2)
#define IVISCOPE_VAL_WIDTH_EITHER                            (3)

    /*- Defined values for IVISCOPE_ATTR_WIDTH_CONDITION -*/
#define IVISCOPE_VAL_WIDTH_WITHIN                           (1)
#define IVISCOPE_VAL_WIDTH_OUTSIDE                           (2)

    /*- Defined values for IVISCOPE_ATTR_AC_LINE_TRIGGER_SLOPE -*/
#define IVISCOPE_VAL_AC_LINE_POSITIVE                        (1)
#define IVISCOPE_VAL_AC_LINE_NEGATIVE                        (2)
#define IVISCOPE_VAL_AC_LINE_EITHER                          (3)

    /*- Defined values for IVISCOPE_ATTR_ACQUISITION_TYPE -*/
#define IVISCOPE_VAL_NORMAL                                 (0)
#define IVISCOPE_VAL_PEAK_DETECT                             (1)
#define IVISCOPE_VAL_HI_RES                                  (2)
```

```
#define IVISCOPE_VAL_ENVELOPE                                   (3)
#define IVISCOPE_VAL_AVERAGE                                    (4)

#define IVISCOPE_VAL_ACQUISITION_TYPE_CLASS_EXT_BASE            (100)
#define IVISCOPE_VAL_ACQUISITION_TYPE_SPECIFIC_EXT_BASE         (1000)

    /*- Defined values for IVISCOPE_ATTR_TRIGGER_MODIFIER -*/
#define IVISCOPE_VAL_NO_TRIGGER_MOD                             (1)
#define IVISCOPE_VAL_AUTO                                       (2)
#define IVISCOPE_VAL_AUTO_LEVEL                                 (3)

#define IVISCOPE_VAL_TRIGGER_MOD_CLASS_EXT_BASE                 (100)
#define IVISCOPE_VAL_TRIGGER_MOD_SPECIFIC_EXT_BASE              (1000)

    /*- Defined values for IVISCOPE_ATTR_SAMPLE_MODE  */
#define IVISCOPE_VAL_REAL_TIME                                  (0)
#define IVISCOPE_VAL_EQUIVALENT_TIME                            (1)

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



   /*  Utility  */

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

   /*- IviScopeBase Capability Group  -*/

ViStatus _VI_FUNC IviScope_ConfigureChannel (ViSession Vi,
                                             ViConstString Channel,
                                             ViReal64 Range,
                                             ViReal64 Offset,
                                             ViInt32 Coupling,
                                             ViReal64 ProbeAttenuation,
                                             ViBoolean Enabled);

ViStatus _VI_FUNC IviScope_ConfigureChanCharacteristics (ViSession Vi,
                                                         ViConstString Channel,
                                                         ViReal64 InputImpedance,
                                                         ViReal64 MaxInputFrequency);

ViStatus _VI_FUNC IviScope_ConfigureAcquisitionType (ViSession Vi,
                                                     ViInt32 AcquisitionType);

ViStatus _VI_FUNC IviScope_ConfigureAcquisitionRecord (ViSession Vi,
                                                       ViReal64 TimePerRecord,
                                                       ViInt32 MinNumPts,
                                                       ViReal64 AcquisitionStartTime);

ViStatus _VI_FUNC IviScope_ActualRecordLength (ViSession Vi,
                                               ViInt32 *ActualRecordLength);

ViStatus _VI_FUNC IviScope_SampleRate (ViSession Vi,
                                       ViReal64 *SampleRate);

ViStatus _VI_FUNC IviScope_ConfigureTrigger (ViSession Vi,
                                             ViInt32 TriggerType,
                                             ViReal64 Holdoff);

ViStatus _VI_FUNC IviScope_ConfigureTriggerCoupling (ViSession Vi,
                                                     ViInt32 Coupling);


ViStatus _VI_FUNC IviScope_ConfigureEdgeTriggerSource (ViSession Vi,
                                                       ViConstString Source,
                                                       ViReal64 Level,
```

```
                                                ViInt32 Slope);

ViStatus _VI_FUNC IviScope_ReadWaveform (ViSession Vi,
                                         ViConstString Channel,
                                         ViInt32 WaveformSize,
                                         ViInt32 MaxTimeMilliseconds,
                                         ViReal64 WaveformArray[],
                                         ViInt32 *ActualPoints,
                                         ViReal64 *InitialX,
                                         ViReal64 *XIncrement);

ViStatus _VI_FUNC IviScope_Abort (ViSession Vi);

ViStatus _VI_FUNC IviScope_InitiateAcquisition (ViSession Vi);

ViStatus _VI_FUNC IviScope_AcquisitionStatus (ViSession Vi,
                                              ViInt32 *Status);

ViStatus _VI_FUNC IviScope_FetchWaveform (ViSession Vi,
                                          ViConstString Channel,
                                          ViInt32 WaveformSize,
                                          ViReal64 WaveformArray[],
                                          ViInt32 *ActualPoints,
                                          ViReal64 *InitialX,
                                          ViReal64 *XIncrement);

ViStatus _VI_FUNC IviScope_IsInvalidWfmElement (ViSession Vi,
                                                ViReal64 ElementValue,
                                                ViBoolean *IsInvalid);

    /*- IviScopeTVTrigger Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureTVTriggerSource (ViSession Vi,
                                                     ViConstString Source,
                                                     ViInt32 TVSignalFormat,
                                                     ViInt32 TVEvent,
                                                     ViInt32 TVPolarity);

ViStatus _VI_FUNC IviScope_ConfigureTVTriggerLineNumber (ViSession Vi,
                                                         ViInt32 TVLineNumber);

    /*- IviScopeRuntTrigger Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureRuntTriggerSource (ViSession Vi,
                                                       ViConstString Source,
                                                       ViReal64 RuntLowThreshold,
                                                       ViReal64 RuntHighThreshold,
                                                       ViInt32 RuntPolarity);

    /*- IviScopeGlitchTrigger Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureGlitchTriggerSource (ViSession Vi,
                                                         ViConstString Source,
                                                         ViReal64 Level,
                                                         ViReal64 GlitchWidth,
                                                         ViInt32 GlitchPolarity,
                                                         ViInt32 GlitchCondition);

    /*- IviScopeWidthTrigger Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureWidthTriggerSource (ViSession Vi,
                                                        ViConstString Source,
                                                        ViReal64 Level,
                                                        ViReal64 WidthLowThreshold,
                                                        ViReal64 WidthHighTreshold,
                                                        ViInt32 WidthPolarity,
                                                        ViInt32 WidthCondition);

    /*- IviScopeAcLineTrigger Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureAcLineTriggerSlope (ViSession Vi,
                                                        ViInt32 ACLineSlope);

    /*- IviScopeTriggerModifier Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureTriggerModifier (ViSession Vi,
                                                     ViInt32 TriggerModifier);
```

```
    /*- IviScopeMinMaxWaveform Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureNumEnvelopes (ViSession Vi,
                                                  ViInt32 NumEnvelopes);

ViStatus _VI_FUNC IviScope_ReadMinMaxWaveform (ViSession Vi,
                                               ViConstString Channel,
                                               ViInt32 WaveformSize,
                                               ViInt32 MaxTimeMilliseconds,
                                               ViReal64 MinWaveform[],
                                               ViReal64 MaxWaveform[],
                                               ViInt32 *ActualPoints,
                                               ViReal64 *InitialX,
                                               ViReal64 *XIncrement);

ViStatus _VI_FUNC IviScope_FetchMinMaxWaveform (ViSession Vi,
                                                ViConstString Channel,
                                                ViInt32 WaveformSize,
                                                ViReal64 MinWaveform[],
                                                ViReal64 MaxWaveform[],
                                                ViInt32 *ActualPoints,
                                                ViReal64 *InitialX,
                                                ViReal64 *XIncrement);

    /*- IviScopeWaveformMeas Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureRefLevels (ViSession Vi,
                                               ViReal64 Low,
                                               ViReal64 Mid,
                                               ViReal64 High);

ViStatus _VI_FUNC IviScope_ReadWaveformMeasurement (ViSession Vi,
                                                    ViConstString Channel
                                                    ViInt32 MeasFunction,
                                                    ViInt32 MaxTimeMilliseconds,
                                                    ViReal64 *Measurement);

ViStatus _VI_FUNC IviScope_FetchWaveformMeasurement (ViSession Vi,
                                                     ViConstString Channel
                                                     ViInt32 MeasFunction,
                                                     ViReal64 *Measurement);

    /*- IviScope Average Acquisition Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureNumAverages (ViSession Vi,
                                                 ViInt32 NumberOfAverages);

    /*- IviScope Continuous Acquisition Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureInitiateContinuous (ViSession Vi,
                                                        ViBoolean ContinuousAcquisition);

    /*- IviScope Interpolation Extension Group -*/
ViStatus _VI_FUNC IviScope_ConfigureInterpolation (ViSession Vi,
                                                   ViInt32 Interpolation);

    /*- IviScope Sample Mode Extension Group -*/
ViStatus _VI_FUNC IviScope_SampleMode (ViSession Vi,
                                       ViInt32 *SampleMode);

    /*- IviScope Probe Auto Sense Extension Group -*/
ViStatus _VI_FUNC IviScope_AutoProbeSenseValue (ViSession Vi,
                                                ViConstString Channel,
                                                ViReal64 *AutoProbeSenseValue);

    /*- IviScope Auto Setup Extension Group -*/
ViStatus _VI_FUNC IviScope_AutoSetup (ViSession Vi);

/**************************************************************************
 *---------------- IviScope Class Error And Completion Codes ----------------*
 **************************************************************************/
#define IVISCOPE_WARN_INVALID_WFM_ELEMENT             (IVI_CLASS_WARN_BASE+0x001)

#define IVISCOPE_ERROR_CHANNEL_NOT_ENABLED            (IVI_CLASS_ERROR_BASE+0x001)
```

```
#define IVISCOPE_ERROR_UNABLE_TO_PERFORM_MEASUREMENT   (IVI_CLASS_ERROR_BASE+0x002)
#define IVISCOPE_ERROR_MAX_TIME_EXCEEDED               (IVI_CLASS_ERROR_BASE+0x003)
#define IVISCOPE_ERROR_INVALID_ACQ_TYPE                (IVI_CLASS_ERROR_BASE+0x004)

/**************************************************************************
 *---------------------------- End Include File ----------------------------*
 **************************************************************************/
#if defined(__cplusplus) || defined(__cplusplus__)
}
#endif
#endif /* IVISCOPE_HEADER */
```

Appendix D. COM IDL File

To ease the development of a compliant IVI-COM driver for the IviScope 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 IIviScope derives from IiviDriver, It is 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 IviScopeTypeLib.idl

```
#if !defined(IVI_SCOPE_TYPELIB_IDL_INCLUDED_)
#define IVI_SCOPE_TYPELIB_IDL_INCLUDED_
/*****************************************************************************
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2002
 *  All rights reserved.
 *
 *
 * FILENAME     : IviScopeTypeLib.idl
 *
 * STATUS       : UN-PUBLISHED.
 * COMPILER     : MSVC++ 6.0, sp4 MIDL
 * CONTENT      : IVI Scope Instrument Class Standard IDL
 *                type library definition
 *
*****************************************************************************/

import "oaidl.idl";
import "ocidl.idl";

[
   uuid(47ed5124-a398-11d4-ba58-000064657374),
   version(1.0),
   helpstring("IviScope 1.0 Type Library")
]
library IviScopeLib
{
   importlib("stdole32.tlb");
   importlib("stdole2.tlb");

#include "IviScope.idl"
};

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

D.2 IviScope.idl

```
#if !defined(IVI_SCOPE_IDL_INCLUDED_)
#define IVI_SCOPE_IDL_INCLUDED_
/*****************************************************************************
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2002
 *  All rights reserved.
 *
 *
 * FILENAME     : IviScope.idl
 *
 * STATUS       : UN-PUBLISHED.
 * COMPILER     : MSVC++ 6.0, sp4 MIDL
```

```
 * CONTENT      : IVI Scope Instrument Class Standard IDL
 *
*****************************************************************************/


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

importlib ("..\TypeLibraries\IviDriverTypeLib.dll");


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

#define HELP_SCOPE(x)  helpstring(HS_SCOPE_ ## x ## ), helpcontext(HC_SCOPE_ ##
x ## )


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

#include "IviScopeEnglish.idl"


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

interface IIviScope;
interface IIviScopeAcquisition;
interface IIviScopeChannels;
interface IIviScopeChannel;
interface IIviScopeReferenceLevel;
interface IIviScopeMeasurements;
interface IIviScopeMeasurement;
interface IIviScopeTrigger;
interface IIviScopeTriggerAcLine;
interface IIviScopeTriggerEdge;
interface IIviScopeTriggerGlitch;
interface IIviScopeTriggerRunt;
interface IIviScopeTriggerTv;
interface IIviScopeTriggerWidth;

#define UUID_IIVI_SCOPE                 47ed524c-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_ACQUISITION     47ed524d-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_CHANNELS        47ed524e-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_CHANNEL         47ed524f-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_REFERENCELEVEL  47ed5250-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_MEASUREMENTS    47ed5251-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_MEASUREMENT     47ed5252-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_TRIGGER         47ed5253-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_TRIGGERACLINE   47ed5254-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_TRIGGEREDGE     47ed5255-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_TRIGGERGLITCH   47ed5256-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_TRIGGERRUNT     47ed5257-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_TRIGGERTV       47ed5258-a398-11d4-ba58-000064657374
#define UUID_IIVI_SCOPE_TRIGGERWIDTH    47ed5259-a398-11d4-ba58-000064657374
```

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

[
   HELP_SCOPE(HRESULTS)
]
typedef enum IviScopeErrorCodesEnum
{
   E_IVISCOPE_CHANNEL_NOT_ENABLED           = MAKE_HRESULT(SEVERITY_ERROR,
FACILITY_ITF, 0x2001),
   E_IVISCOPE_UNABLE_TO_PERFORM_MEASUREMENT  = MAKE_HRESULT(SEVERITY_ERROR,
FACILITY_ITF, 0x2002),
   E_IVISCOPE_MAX_TIME_EXCEEDED             = MAKE_HRESULT(SEVERITY_ERROR,
FACILITY_ITF, 0x2003),
   E_IVISCOPE_INVALID_ACQ_TYPE              = MAKE_HRESULT(SEVERITY_ERROR,
FACILITY_ITF, 0x2004),
   S_IVISCOPE_INVALID_WFM_ELEMENT           = MAKE_HRESULT(SEVERITY_SUCCESS,
FACILITY_ITF, 0x2001)
} IviScopeErrorCodesEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(AC_LINE_SLOPE_ENUM)
]
enum IviScopeACLineSlopeEnum
{
   IviScopeACLinePositive                    = 1,
   IviScopeACLineNegative                    = 2,
   IviScopeACLineEither                      = 3
} IviScopeACLineSlopeEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(ACQUISITION_STATUS_ENUM)
]
enum IviScopeAcquisitionStatusEnum
{
   IviScopeAcquisitionStatusComplete         = 1,
   IviScopeAcquisitionStatusInProgress       = 0,
   IviScopeAcquisitionStatusUnknown          = -1
} IviScopeAcquisitionStatusEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(ACQUISITION_TYPE_ENUM)
]
enum IviScopeAcquisitionTypeEnum
{
   IviScopeAcquisitionTypeNormal             = 0,
   IviScopeAcquisitionTypePeakDetect         = 1,
   IviScopeAcquisitionTypeHiRes              = 2,
   IviScopeAcquisitionTypeEnvelope           = 3,
   IviScopeAcquisitionTypeAverage            = 4
```

```
} IviScopeAcquisitionTypeEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(GLITCH_CONDITION_ENUM)
]
enum IviScopeGlitchConditionEnum
{
   IviScopeGlitchLessThan                    = 1,
   IviScopeGlitchGreaterThan                 = 2
} IviScopeGlitchConditionEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(GLITCH_POLARITY_ENUM)
]
enum IviScopeGlitchPolarityEnum
{
   IviScopeGlitchPositive                    = 1,
   IviScopeGlitchNegative                    = 2,
   IviScopeGlitchEither                      = 3
} IviScopeGlitchPolarityEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(INTERPOLATION_ENUM)
]
enum IviScopeInterpolationEnum
{
   IviScopeInterpolationNone                 = 1,
   IviScopeInterpolationSineX                = 2,
   IviScopeInterpolationLinear               = 3
} IviScopeInterpolationEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(MEASUREMENT_ENUM)
]
enum IviScopeMeasurementEnum
{
   IviScopeMeasurementRiseTime               = 0,
   IviScopeMeasurementFallTime               = 1,
   IviScopeMeasurementFrequency              = 2,
   IviScopeMeasurementPeriod                 = 3,
   IviScopeMeasurementVoltageRMS             = 4,
   IviScopeMeasurementVoltagePeakToPeak      = 5,
   IviScopeMeasurementVoltageMax             = 6,
   IviScopeMeasurementVoltageMin             = 7,
   IviScopeMeasurementVoltageHigh            = 8,
   IviScopeMeasurementVoltageLow             = 9,
   IviScopeMeasurementVoltageAverage         = 10,
```

```
   IviScopeMeasurementWidthNeg               = 11,
   IviScopeMeasurementWidthPos               = 12,
   IviScopeMeasurementDutyCycleNeg           = 13,
   IviScopeMeasurementDutyCyclePos           = 14,
   IviScopeMeasurementAmplitude              = 15,
   IviScopeMeasurementVoltageCycleRMS        = 16,
   IviScopeMeasurementVoltageCycleAverage    = 17,
   IviScopeMeasurementOvershoot              = 18,
   IviScopeMeasurementPreShoot               = 19
} IviScopeMeasurementEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(RUNT_POLARITY_ENUM)
]
enum IviScopeRuntPolarityEnum
{
   IviScopeRuntPositive                      = 1,
   IviScopeRuntNegative                      = 2,
   IviScopeRuntEither                        = 3
} IviScopeRuntPolarityEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(SAMPLE_MODE_ENUM)
]
enum IviScopeSampleModeEnum
{
   IviScopeSampleModeRealTime                = 0,
   IviScopeSampleModeEquivalentTime          = 1
} IviScopeSampleModeEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TIME_OUT_ENUM)
]
enum IviScopeTimeOutEnum
{
   IviScopeTimeOutInfinite                   = 0xFFFFFFFFUL,
   IviScopeTimeOutImmediate                  = 0
} IviScopeTimeOutEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TRIGGER_COUPLING_ENUM)
]
enum IviScopeTriggerCouplingEnum
{
   IviScopeTriggerCouplingAC                 = 0,
   IviScopeTriggerCouplingDC                 = 1,
   IviScopeTriggerCouplingHFReject           = 3,
```

```
   IviScopeTriggerCouplingLFReject           = 4,
   IviScopeTriggerCouplingNoiseReject        = 5
} IviScopeTriggerCouplingEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TRIGGER_MODIFIER_ENUM)
]
enum IviScopeTriggerModifierEnum
{
   IviScopeTriggerModifierNone               = 1,
   IviScopeTriggerModifierAuto               = 2,
   IviScopeTriggerModifierAutoLevel          = 3
} IviScopeTriggerModifierEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TRIGGER_SLOPE_ENUM)
]
enum IviScopeTriggerSlopeEnum
{
   IviScopeTriggerSlopePositive              = 1,
   IviScopeTriggerSlopeNegative              = 0
} IviScopeTriggerSlopeEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TRIGGER_TYPE_ENUM)
]
enum IviScopeTriggerTypeEnum
{
   IviScopeTriggerEdge                       = 1,
   IviScopeTriggerWidth                      = 2,
   IviScopeTriggerRunt                       = 3,
   IviScopeTriggerGlitch                     = 4,
   IviScopeTriggerTV                         = 5,
   IviScopeTriggerImmediate                  = 6,
   IviScopeTriggerACLine                     = 7
} IviScopeTriggerTypeEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TV_SIGNAL_FORMAT_ENUM)
]
enum IviScopeTVSignalFormatEnum
{
   IviScopeTVSignalFormatNTSC                = 1,
   IviScopeTVSignalFormatPAL                 = 2,
   IviScopeTVSignalFormatSECAM               = 3
} IviScopeTVSignalFormatEnum;
```

```
typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TV_TRIGGER_EVENT_ENUM)
]
enum IviScopeTVTriggerEventEnum
{
   IviScopeTVEventField1                        = 1,
   IviScopeTVEventField2                        = 2,
   IviScopeTVEventAnyField                      = 3,
   IviScopeTVEventAnyLine                       = 4,
   IviScopeTVEventLineNumber                    = 5
} IviScopeTVTriggerEventEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(TV_TRIGGER_POLARITY_ENUM)
]
enum IviScopeTVTriggerPolarityEnum
{
   IviScopeTVPositive                           = 1,
   IviScopeTVNegative                           = 2
} IviScopeTVTriggerPolarityEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(VERTICAL_COUPLING_ENUM)
]
enum IviScopeVerticalCouplingEnum
{
   IviScopeVerticalCouplingAC                   = 0,
   IviScopeVerticalCouplingDC                   = 1,
   IviScopeVerticalCouplingGnd                  = 2
} IviScopeVerticalCouplingEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(WIDTH_CONDITION_ENUM)
]
enum IviScopeWidthConditionEnum
{
   IviScopeWidthWithin                          = 1,
   IviScopeWidthOutside                         = 2
} IviScopeWidthConditionEnum;


typedef
[
   public,
   v1_enum,
   HELP_SCOPE(WIDTH_POLARITY_ENUM)
]
```

```
enum IviScopeWidthPolarityEnum
{
   IviScopeWidthPositive                     = 1,
   IviScopeWidthNegative                     = 2,
   IviScopeWidthEither                       = 3
} IviScopeWidthPolarityEnum;



//---------------------------------------------------------------------------
//   IVI Scope Driver Root Level Interface
//---------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE),
   HELP_SCOPE(I_IVI_SCOPE),
   oleautomation,
   pointer_default(unique)
]
interface IIviScope : IIviDriver
{
//----------------------------------------- Acquisition Interface Reference
   [ propget, HELP_SCOPE(ACQUISITION) ]
   HRESULT Acquisition ([out, retval] IIviScopeAcquisition **pVal);


//----------------------------------------- Channels Interface Reference
   [ propget, HELP_SCOPE(CHANNELS) ]
   HRESULT Channels ([out, retval] IIviScopeChannels **pVal);


//----------------------------------------- Measurements Interface Reference
   [ propget, HELP_SCOPE(MEASUREMENTS) ]
   HRESULT Measurements ([out, retval] IIviScopeMeasurements **pVal);


//----------------------------------------- ReferenceLevel Interface Reference
   [ propget, HELP_SCOPE(REFERENCE_LEVEL) ]
   HRESULT ReferenceLevel ([out, retval] IIviScopeReferenceLevel **pVal);


//----------------------------------------- Trigger Interface Reference
   [ propget, HELP_SCOPE(TRIGGER) ]
   HRESULT Trigger ([out, retval] IIviScopeTrigger **pVal);

};


//---------------------------------------------------------------------------
//   Acquisition Interface
//---------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_ACQUISITION),
   HELP_SCOPE(I_IVI_SCOPE_ACQUISITION),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeAcquisition : IUnknown
{
//----------------------------------------- ConfigureRecord()
```

```
   [ HELP_SCOPE(CONFIGURE_RECORD) ]
   HRESULT ConfigureRecord(
         [in] DOUBLE TimePerRecord,
         [in] LONG NumberOfPointsMin,
         [in] DOUBLE StartTime);


//---------------------------------------- NumberOfAverages
   [ propget, HELP_SCOPE(AVERAGES) ]
   HRESULT NumberOfAverages([out, retval] LONG *pVal);

   [ propput, HELP_SCOPE(AVERAGES) ]
   HRESULT NumberOfAverages([in] LONG newVal);


//---------------------------------------- NumberOfEnvelopes
   [ propget, HELP_SCOPE(ENVELOPES) ]
   HRESULT NumberOfEnvelopes([out, retval] LONG *pVal);

   [ propput, HELP_SCOPE(ENVELOPES) ]
   HRESULT NumberOfEnvelopes([in] LONG newVal);


//---------------------------------------- Interpolation
   [ propget, HELP_SCOPE(INTERPOLATION) ]
   HRESULT Interpolation([out, retval] IviScopeInterpolationEnum *pVal);

   [ propput, HELP_SCOPE(INTERPOLATION) ]
   HRESULT Interpolation([in] IviScopeInterpolationEnum newVal);


//---------------------------------------- RecordLength
   [ propget, HELP_SCOPE(RECORD_LENGTH) ]
   HRESULT RecordLength([out, retval] LONG* pVal);


//---------------------------------------- SampleMode
   [ propget, HELP_SCOPE(SAMPLE_MODE) ]
   HRESULT SampleMode([out, retval] IviScopeSampleModeEnum *pVal);


//---------------------------------------- SampleRate
   [ propget, HELP_SCOPE(SAMPLE_RATE) ]
   HRESULT SampleRate([out, retval] DOUBLE *pVal);


//---------------------------------------- Type
   [ propget, HELP_SCOPE(TYPE) ]
   HRESULT Type([out, retval] IviScopeAcquisitionTypeEnum *pVal);

   [ propput, HELP_SCOPE(TYPE) ]
   HRESULT Type([in] IviScopeAcquisitionTypeEnum newVal);


//---------------------------------------- NumberOfPointsMin
   [ propget, HELP_SCOPE(NUMBER_OF_POINTS_MIN) ]
   HRESULT NumberOfPointsMin([out, retval] LONG *pVal);

   [ propput, HELP_SCOPE(NUMBER_OF_POINTS_MIN) ]
   HRESULT NumberOfPointsMin([in] LONG newVal);


//---------------------------------------- StartTime
```

```
   [ propget, HELP_SCOPE(START_TIME) ]
   HRESULT StartTime([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(START_TIME) ]
   HRESULT StartTime([in] DOUBLE newVal);


//---------------------------------------- TimePerRecord
   [ propget, HELP_SCOPE(TIME_PER_RECORD) ]
   HRESULT TimePerRecord([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(TIME_PER_RECORD) ]
   HRESULT TimePerRecord([in] DOUBLE newVal);

};


//------------------------------------------------------------------------------
//   Reference Level Interface
//------------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_REFERENCELEVEL),
   HELP_SCOPE(I_IVI_SCOPE_REFERENCE_LEVEL),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeReferenceLevel : IUnknown
{
//---------------------------------------- Configure()
   [ HELP_SCOPE(CONFIGURE_REFERENCE_LEVELS) ]
   HRESULT Configure(
         [in] DOUBLE Low,
         [in] DOUBLE Mid,
         [in] DOUBLE High);


//---------------------------------------- High
   [ propget, HELP_SCOPE(REFERENCE_HIGH) ]
   HRESULT High([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(REFERENCE_HIGH) ]
   HRESULT High([in] DOUBLE newVal);


//---------------------------------------- Low
   [ propget, HELP_SCOPE(REFERENCE_LOW) ]
   HRESULT Low([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(REFERENCE_LOW) ]
   HRESULT Low([in] DOUBLE newVal);


//---------------------------------------- Mid
   [ propget, HELP_SCOPE(REFERENCE_MID) ]
   HRESULT Mid([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(REFERENCE_MID) ]
   HRESULT Mid([in] DOUBLE newVal);

};
```

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

[
   object,
   uuid(UUID_IIVI_SCOPE_CHANNELS),
   HELP_SCOPE(I_IVI_SCOPE_CHANNELS),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeChannels : IUnknown
{
//---------------------------------------- Item
   [ propget, HELP_SCOPE(CHANNELS_ITEM) ]
   HRESULT Item (
         [in] BSTR Name,
         [out, retval] IIviScopeChannel **pVal);


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


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

};


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

[
   object,
   uuid(UUID_IIVI_SCOPE_CHANNEL),
   HELP_SCOPE(I_IVI_SCOPE_CHANNEL),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeChannel : IUnknown
{
//---------------------------------------- Configure()
   [ HELP_SCOPE(CONFIGURE_CHANNEL) ]
   HRESULT Configure(
         [in] DOUBLE Range,
         [in] DOUBLE Offset,
         [in] IviScopeVerticalCouplingEnum Coupling,
         [in] DOUBLE ProbeAttenuation,
         [in] VARIANT_BOOL Enabled);


//---------------------------------------- ConfigureCharacteristics()
   [ HELP_SCOPE(CONFIGURE_CHARACTERISTICS) ]
   HRESULT ConfigureCharacteristics(
         [in] DOUBLE InputImpedance,
         [in] DOUBLE InputFrequencyMax);
```

```
//------------------------------------------ ProbeSense
   [ propget, HELP_SCOPE(PROBE_SENSE) ]
   HRESULT ProbeSense([out, retval] DOUBLE *pVal);


//------------------------------------------ Coupling
   [ propget, HELP_SCOPE(COUPLING) ]
   HRESULT Coupling([out, retval] IviScopeVerticalCouplingEnum *pVal);

   [ propput, HELP_SCOPE(COUPLING) ]
   HRESULT Coupling([in] IviScopeVerticalCouplingEnum newVal);


//------------------------------------------ Enabled
   [ propget, HELP_SCOPE(ENABLED) ]
   HRESULT Enabled([out, retval] VARIANT_BOOL *pVal);

   [ propput, HELP_SCOPE(ENABLED) ]
   HRESULT Enabled([in] VARIANT_BOOL newVal);


//------------------------------------------ InputFrequencyMax
   [ propget, HELP_SCOPE(INPUT_FREQUENCY_MAX) ]
   HRESULT InputFrequencyMax([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(INPUT_FREQUENCY_MAX) ]
   HRESULT InputFrequencyMax([in] DOUBLE newVal);


//------------------------------------------ InputImpedance
   [ propget, HELP_SCOPE(INPUT_IMPEDANCE) ]
   HRESULT InputImpedance( [out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(INPUT_IMPEDANCE) ]
   HRESULT InputImpedance([in] DOUBLE newVal);


//------------------------------------------ Offset
   [ propget, HELP_SCOPE(OFFSET) ]
   HRESULT Offset([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(OFFSET) ]
   HRESULT Offset([in] DOUBLE newVal);


//------------------------------------------ ProbeAttenuation
   [ propget, HELP_SCOPE(PROBE_ATTENUATION) ]
   HRESULT ProbeAttenuation([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(PROBE_ATTENUATION) ]
   HRESULT ProbeAttenuation([in] DOUBLE newVal);


//------------------------------------------ Range
   [ propget, HELP_SCOPE(RANGE) ]
   HRESULT Range([out, retval] DOUBLE *pVal);

   [ propput, HELP_SCOPE(RANGE) ]
   HRESULT Range([in] DOUBLE newVal);

};
```

```
//-----------------------------------------------------------------------------
//   Measurements Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_MEASUREMENTS),
   HELP_SCOPE(I_IVI_SCOPE_MEASUREMENTS),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeMeasurements : IUnknown
{
//---------------------------------------- Item
   [ propget, HELP_SCOPE(MEASUREMENTS_ITEM) ]
   HRESULT Item (
         [in] BSTR Name,
         [out, retval] IIviScopeMeasurement **pVal);


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


//---------------------------------------- Name
   [ propget, HELP_SCOPE(MEASUREMENTS_NAME) ]
   HRESULT Name ([in] LONG Index, [out, retval] BSTR *pVal);


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


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


//---------------------------------------- AutoSetup
   [ HELP_SCOPE(AUTO_SETUP) ]
   HRESULT AutoSetup();


//---------------------------------------- Status
   [ HELP_SCOPE(STATUS) ]
   HRESULT Status([out, retval] IviScopeAcquisitionStatusEnum *Status);


//---------------------------------------- IsWaveformElementInvalid()
   [ HELP_SCOPE(IS_WAVEFORM_ELEMENT_INVALID) ]
   HRESULT IsWaveformElementInvalid(
         [in] DOUBLE Element,
         [out, retval] VARIANT_BOOL* IsInvalid);

};


//-----------------------------------------------------------------------------
//   Measurement Interface
//-----------------------------------------------------------------------------
```

```
[
   object,
   uuid(UUID_IIVI_SCOPE_MEASUREMENT),
   HELP_SCOPE(I_IVI_SCOPE_MEASUREMENT),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeMeasurement : IUnknown
{
//---------------------------------------- FetchWaveform()
   [ HELP_SCOPE(FETCH_WAVEFORM) ]
   HRESULT FetchWaveform(
         [in,out] SAFEARRAY (DOUBLE) *WaveformArray,
         [in,out] DOUBLE *InitialX,
         [in,out] DOUBLE *XIncrement);


//---------------------------------------- FetchWaveformMeasurement()
   [ HELP_SCOPE(FETCH_WAVEFORM_MEASUREMENT) ]
   HRESULT FetchWaveformMeasurement(
         [in] IviScopeMeasurementEnum MeasFunction,
         [in,out] DOUBLE *Measurement);


//---------------------------------------- FetchWaveformMinMax()
   [ HELP_SCOPE(FETCH_WAVEFORM_MIN_MAX) ]
   HRESULT FetchWaveformMinMax(
         [in,out] SAFEARRAY (DOUBLE) *MinWaveform,
         [in,out] SAFEARRAY (DOUBLE) *MaxWaveform,
         [in,out] DOUBLE *InitialX,
         [in,out] DOUBLE *XIncrement);


//---------------------------------------- ReadWaveform()
   [ HELP_SCOPE(READ_WAVEFORM) ]
   HRESULT ReadWaveform(
         [in] LONG MaxTimeMilliseconds,
         [in,out] SAFEARRAY (DOUBLE) *WaveformArray,
         [in,out] DOUBLE *InitialX,
         [in,out] DOUBLE *XIncrement);


//---------------------------------------- ReadWaveformMeasurement()
   [ HELP_SCOPE(READ_WAVEFORM_MEASUREMENT) ]
   HRESULT ReadWaveformMeasurement(
         [in] IviScopeMeasurementEnum MeasFunction,
         [in] LONG MaxTimeMilliseconds,
         [in,out] DOUBLE *Measurement);


//---------------------------------------- ReadWaveformMinMax()
   [ HELP_SCOPE(READ_WAVEFORM_MIN_MAX) ]
   HRESULT ReadWaveformMinMax(
         [in] LONG MaxTimeMilliseconds,
         [in,out] SAFEARRAY (DOUBLE) *MinWaveform,
         [in,out] SAFEARRAY (DOUBLE) *MaxWaveform,
         [in,out] DOUBLE *InitialX,
         [in,out] DOUBLE *XIncrement);

};


//-----------------------------------------------------------------------------
```

```
//   Trigger Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_TRIGGER),
   HELP_SCOPE(I_IVI_SCOPE_TRIGGER),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeTrigger : IUnknown
{
//---------------------------------------- Configure
   [ HELP_SCOPE(CONFIGURE) ]
   HRESULT Configure(
         [in] IviScopeTriggerTypeEnum Type,
         [in] DOUBLE Holdoff);


//---------------------------------------- Continuous
   [ propput, HELP_SCOPE(CONTINUOUS) ]
   HRESULT Continuous([in] VARIANT_BOOL newVal);

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


//---------------------------------------- Coupling
   [ propput, HELP_SCOPE(TRIGGER_COUPLING) ]
   HRESULT Coupling([in] IviScopeTriggerCouplingEnum newVal);

   [ propget, HELP_SCOPE(TRIGGER_COUPLING) ]
   HRESULT Coupling([out, retval] IviScopeTriggerCouplingEnum *pVal);


//---------------------------------------- Level
   [ propput, HELP_SCOPE(LEVEL) ]
   HRESULT Level([in] DOUBLE newVal);

   [ propget, HELP_SCOPE(LEVEL) ]
   HRESULT Level([out, retval] DOUBLE *pVal);


//---------------------------------------- Modifier
   [ propput, HELP_SCOPE(MODIFIER) ]
   HRESULT Modifier([in] IviScopeTriggerModifierEnum newVal);

   [ propget, HELP_SCOPE(MODIFIER) ]
   HRESULT Modifier([out, retval] IviScopeTriggerModifierEnum *pVal);


//---------------------------------------- Source
   [ propput, HELP_SCOPE(SOURCE) ]
   HRESULT Source([in] BSTR newVal);

   [ propget, HELP_SCOPE(SOURCE) ]
   HRESULT Source([out, retval] BSTR *pVal);


//---------------------------------------- Holdoff
   [ propput, HELP_SCOPE(HOLDOFF) ]
   HRESULT Holdoff([in] DOUBLE newVal);
```

```
   [ propget, HELP_SCOPE(HOLDOFF) ]
   HRESULT Holdoff([out, retval] DOUBLE *pVal);


//----------------------------------------- Type
   [ propput, HELP_SCOPE(TRIGGER_TYPE) ]
   HRESULT Type([in] IviScopeTriggerTypeEnum newVal);

   [ propget, HELP_SCOPE(TRIGGER_TYPE) ]
   HRESULT Type([out, retval] IviScopeTriggerTypeEnum *pVal);


//----------------------------------------- AcLine Interface Reference
   [ propget, HELP_SCOPE(TRIGGER_AC_LINE) ]
   HRESULT AcLine([out, retval] IIviScopeTriggerAcLine **pVal);


//----------------------------------------- Edge Interface Reference
   [ propget, HELP_SCOPE(TRIGGER_EDGE) ]
   HRESULT Edge([out, retval] IIviScopeTriggerEdge **pVal);


//----------------------------------------- Glitch Interface Reference
   [ propget, HELP_SCOPE(TRIGGER_GLITCH) ]
   HRESULT Glitch([out, retval] IIviScopeTriggerGlitch **pVal);


//----------------------------------------- Runt Interface Reference
   [ propget, HELP_SCOPE(TRIGGER_RUNT) ]
   HRESULT Runt([out, retval] IIviScopeTriggerRunt **pVal);


//----------------------------------------- TV Interface Reference
   [ propget, HELP_SCOPE(TRIGGER_TV) ]
   HRESULT TV([out, retval] IIviScopeTriggerTv **pVal);


//----------------------------------------- Width Interface Reference
   [ propget, HELP_SCOPE(TRIGGER_WIDTH) ]
   HRESULT Width([out, retval] IIviScopeTriggerWidth **pVal);

};


//-----------------------------------------------------------------------------
//   AcLine Trigger Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_TRIGGERACLINE),
   HELP_SCOPE(I_IVI_SCOPE_TRIGGER_AC_LINE),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeTriggerAcLine : IUnknown
{
//----------------------------------------- Slope
   [ propput, HELP_SCOPE(AC_LINE_SLOPE) ]
   HRESULT Slope([in] IviScopeACLineSlopeEnum newVal);

   [ propget, HELP_SCOPE(AC_LINE_SLOPE) ]
   HRESULT Slope([out, retval] IviScopeACLineSlopeEnum *pVal);
```

```
};


//-----------------------------------------------------------------------------
//   Edge Trigger Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_TRIGGEREDGE),
   HELP_SCOPE(I_IVI_SCOPE_TRIGGER_EDGE),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeTriggerEdge : IUnknown
{
//---------------------------------------- Configure
   [ HELP_SCOPE(CONFIGURE_EDGE_SOURCE) ]
   HRESULT Configure(
         [in] BSTR Source,
         [in] DOUBLE Level,
         [in] IviScopeTriggerSlopeEnum Slope);


//---------------------------------------- Slope
   [ propput, HELP_SCOPE(SLOPE) ]
   HRESULT Slope([in] IviScopeTriggerSlopeEnum newVal);

   [ propget, HELP_SCOPE(SLOPE) ]
   HRESULT Slope([out, retval] IviScopeTriggerSlopeEnum *pVal);

};


//-----------------------------------------------------------------------------
//   Glitch Trigger Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_TRIGGERGLITCH),
   HELP_SCOPE(I_IVI_SCOPE_TRIGGER_GLITCH),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeTriggerGlitch : IUnknown
{
//---------------------------------------- Configure
   [ HELP_SCOPE(GLITCH_CONFIGURE_TRIGGER) ]
   HRESULT Configure(
         [in] BSTR Source,
         [in] DOUBLE Level,
         [in] DOUBLE Width,
         [in] IviScopeGlitchPolarityEnum polarity,
         [in] IviScopeGlitchConditionEnum condition);


//---------------------------------------- Condition
   [ propput, HELP_SCOPE(GLITCH_CONDITION) ]
   HRESULT Condition([in] IviScopeGlitchConditionEnum newVal);

   [ propget, HELP_SCOPE(GLITCH_CONDITION) ]
```

```
   HRESULT Condition([out, retval] IviScopeGlitchConditionEnum *pVal);


//---------------------------------------- Polarity
   [ propput, HELP_SCOPE(GLITCH_POLARITY) ]
   HRESULT Polarity([in] IviScopeGlitchPolarityEnum newVal);

   [ propget, HELP_SCOPE(GLITCH_POLARITY) ]
   HRESULT Polarity([out, retval] IviScopeGlitchPolarityEnum *pVal);


//---------------------------------------- Width
   [ propput, HELP_SCOPE(GLITCH_WIDTH) ]
   HRESULT Width([in] DOUBLE newVal);

   [ propget, HELP_SCOPE(GLITCH_WIDTH) ]
   HRESULT Width([out, retval] DOUBLE *pVal);

};


//------------------------------------------------------------------------------
//   Runt Trigger Interface
//------------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_TRIGGERRUNT),
   HELP_SCOPE(I_IVI_SCOPE_TRIGGER_RUNT),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeTriggerRunt : IUnknown
{
//---------------------------------------- Configure
   [ HELP_SCOPE(RUNT_CONFIGURE_TRIGGER) ]
   HRESULT Configure(
         [in] BSTR Source,
         [in] DOUBLE ThresholdLow,
         [in] DOUBLE ThresholdHigh,
         [in] IviScopeRuntPolarityEnum Polarity);


//---------------------------------------- Polarity
   [ propput, HELP_SCOPE(RUNT_POLARITY) ]
   HRESULT Polarity([in] IviScopeRuntPolarityEnum newVal);

   [ propget, HELP_SCOPE(RUNT_POLARITY) ]
   HRESULT Polarity([out, retval] IviScopeRuntPolarityEnum *pVal);


//---------------------------------------- ThresholdHigh
   [ propput, HELP_SCOPE(RUNT_THRESHOLD_HIGH) ]
   HRESULT ThresholdHigh([in] DOUBLE newVal);

   [ propget, HELP_SCOPE(RUNT_THRESHOLD_HIGH) ]
   HRESULT ThresholdHigh([out, retval] DOUBLE *pVal);


//---------------------------------------- ThresholdLow
   [ propput, HELP_SCOPE(RUNT_THRESHOLD_LOW) ]
   HRESULT ThresholdLow([in] DOUBLE newVal);
```

```
   [ propget, HELP_SCOPE(RUNT_THRESHOLD_LOW) ]
   HRESULT ThresholdLow([out, retval] DOUBLE *pVal);

};


//-----------------------------------------------------------------------------
//   TV Trigger Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_TRIGGERTV),
   HELP_SCOPE(I_IVI_SCOPE_TRIGGER_TV),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeTriggerTv : IUnknown
{
//---------------------------------------- Configure
   [ HELP_SCOPE(TV_CONFIGURE_TRIGGER) ]
   HRESULT Configure(
         [in] BSTR Source,
         [in] IviScopeTVSignalFormatEnum SignalFormat,
         [in] IviScopeTVTriggerEventEnum Event,
         [in] IviScopeTVTriggerPolarityEnum Polarity);


//---------------------------------------- LineNumber
   [ propput, HELP_SCOPE(TV_LINE_NUMBER) ]
   HRESULT LineNumber([in] LONG newVal);

   [ propget, HELP_SCOPE(TV_LINE_NUMBER) ]
   HRESULT LineNumber([out, retval] LONG *pVal);


//---------------------------------------- Event
   [ propput, HELP_SCOPE(TV_EVENT) ]
   HRESULT Event([in] IviScopeTVTriggerEventEnum newVal);

   [ propget, HELP_SCOPE(TV_EVENT) ]
   HRESULT Event([out, retval] IviScopeTVTriggerEventEnum *pVal);


//---------------------------------------- Polarity
   [ propput, HELP_SCOPE(TV_POLARITY) ]
   HRESULT Polarity([in] IviScopeTVTriggerPolarityEnum newVal);

   [ propget, HELP_SCOPE(TV_POLARITY) ]
   HRESULT Polarity([out, retval] IviScopeTVTriggerPolarityEnum *pVal);


//---------------------------------------- SignalFormat
   [ propput, HELP_SCOPE(TV_SIGNAL_FORMAT) ]
   HRESULT SignalFormat([in] IviScopeTVSignalFormatEnum newVal);

   [ propget, HELP_SCOPE(TV_SIGNAL_FORMAT) ]
   HRESULT SignalFormat([out, retval] IviScopeTVSignalFormatEnum *pVal);

};


//-----------------------------------------------------------------------------
```

```
//   Width Trigger Interface
//-----------------------------------------------------------------------------

[
   object,
   uuid(UUID_IIVI_SCOPE_TRIGGERWIDTH),
   HELP_SCOPE(I_IVI_SCOPE_TRIGGER_WIDTH),
   oleautomation,
   pointer_default(unique)
]
interface IIviScopeTriggerWidth : IUnknown
{
//--------------------------------------- Configure
   [ HELP_SCOPE(WIDTH_CONFIGURE_TRIGGER) ]
   HRESULT Configure(
         [in] BSTR Source,
         [in] DOUBLE Level,
         [in] DOUBLE ThresholdLow,
         [in] DOUBLE ThresholdHigh,
         [in] IviScopeWidthPolarityEnum Polarity,
         [in] IviScopeWidthConditionEnum Condition);


//--------------------------------------- Condition
   [ propput, HELP_SCOPE(WIDTH_CONDITION) ]
   HRESULT Condition([in] IviScopeWidthConditionEnum newVal);

   [ propget, HELP_SCOPE(WIDTH_CONDITION) ]
   HRESULT Condition([out, retval] IviScopeWidthConditionEnum *pVal);


//--------------------------------------- Polarity
   [ propput, HELP_SCOPE(WIDTH_POLARITY) ]
   HRESULT Polarity([in] IviScopeWidthPolarityEnum newVal);

   [ propget, HELP_SCOPE(WIDTH_POLARITY) ]
   HRESULT Polarity([out, retval] IviScopeWidthPolarityEnum *pVal);


//--------------------------------------- ThresholdLow
   [ propput, HELP_SCOPE(WIDTH_THRESHOLD_LOW) ]
   HRESULT ThresholdLow([in] DOUBLE newVal);

   [ propget, HELP_SCOPE(WIDTH_THRESHOLD_LOW) ]
   HRESULT ThresholdLow([out, retval] DOUBLE *pVal);


//--------------------------------------- ThresholdHigh
   [ propput, HELP_SCOPE(WIDTH_THRESHOLD_HIGH) ]
   HRESULT ThresholdHigh([in] DOUBLE newVal);

   [ propget, HELP_SCOPE(WIDTH_THRESHOLD_HIGH) ]
   HRESULT ThresholdHigh([out, retval] DOUBLE *pVal);

};
#endif // !defined(IVI_SCOPE_IDL_INCLUDED_)
```

D.3 IviScopeEnglish.idl

```
#if !defined(IVI_SCOPE_IDL_ENGLISH_INCLUDED_)
#define IVI_SCOPE_IDL_ENGLISH_INCLUDED_
/*****************************************************************************
 *
```

```
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001
 *  All rights reserved.
 *
 *
 * FILENAME     : IviScopeEnglish.idl
 *
 * STATUS       : UN-PUBLISHED.
 * COMPILER     : MSVC++ 6.0, sp4 MIDL
 * CONTENT      : IVI Scope Instrument Class Standard IDL
 *                help context IDs and help strings
 *
*****************************************************************************/


#define HC_SCOPE_BASE  400


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

#define HC_SCOPE_HRESULTS                               HC_SCOPE_BASE + 0

#define HC_SCOPE_AC_LINE_SLOPE_ENUM                     HC_SCOPE_BASE + 1
#define HC_SCOPE_ACQUISITION_STATUS_ENUM                HC_SCOPE_BASE + 2
#define HC_SCOPE_ACQUISITION_TYPE_ENUM                  HC_SCOPE_BASE + 3
#define HC_SCOPE_GLITCH_CONDITION_ENUM                  HC_SCOPE_BASE + 4
#define HC_SCOPE_GLITCH_POLARITY_ENUM                   HC_SCOPE_BASE + 5
#define HC_SCOPE_INTERPOLATION_ENUM                     HC_SCOPE_BASE + 6
#define HC_SCOPE_MEASUREMENT_ENUM                       HC_SCOPE_BASE + 7
#define HC_SCOPE_RUNT_POLARITY_ENUM                     HC_SCOPE_BASE + 8
#define HC_SCOPE_SAMPLE_MODE_ENUM                       HC_SCOPE_BASE + 9
#define HC_SCOPE_TIME_OUT_ENUM                          HC_SCOPE_BASE + 10
#define HC_SCOPE_TRIGGER_COUPLING_ENUM                  HC_SCOPE_BASE + 11
#define HC_SCOPE_TRIGGER_MODIFIER_ENUM                  HC_SCOPE_BASE + 12
#define HC_SCOPE_TRIGGER_SLOPE_ENUM                     HC_SCOPE_BASE + 13
#define HC_SCOPE_TRIGGER_TYPE_ENUM                      HC_SCOPE_BASE + 14
#define HC_SCOPE_TV_SIGNAL_FORMAT_ENUM                  HC_SCOPE_BASE + 15
#define HC_SCOPE_TV_TRIGGER_EVENT_ENUM                  HC_SCOPE_BASE + 16
#define HC_SCOPE_TV_TRIGGER_POLARITY_ENUM               HC_SCOPE_BASE + 17
#define HC_SCOPE_VERTICAL_COUPLING_ENUM                 HC_SCOPE_BASE + 18
#define HC_SCOPE_WIDTH_CONDITION_ENUM                   HC_SCOPE_BASE + 19
#define HC_SCOPE_WIDTH_POLARITY_ENUM                    HC_SCOPE_BASE + 20


//---------------------------------------------------------------------------
//   IVI Scope Driver Root Level Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE                            HC_SCOPE_BASE + 21
#define HC_SCOPE_ACQUISITION                            HC_SCOPE_BASE + 22
#define HC_SCOPE_CHANNELS                               HC_SCOPE_BASE + 23
#define HC_SCOPE_MEASUREMENTS                           HC_SCOPE_BASE + 24
#define HC_SCOPE_REFERENCE_LEVEL                        HC_SCOPE_BASE + 25
#define HC_SCOPE_TRIGGER                                HC_SCOPE_BASE + 26

//---------------------------------------------------------------------------
//   IIviScopeAcquisition Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_ACQUISITION                HC_SCOPE_BASE + 27
#define HC_SCOPE_CONFIGURE_RECORD                       HC_SCOPE_BASE + 28
#define HC_SCOPE_AVERAGES                               HC_SCOPE_BASE + 29
```

```
#define HC_SCOPE_ENVELOPES                              HC_SCOPE_BASE + 30
#define HC_SCOPE_INTERPOLATION                          HC_SCOPE_BASE + 31
#define HC_SCOPE_RECORD_LENGTH                          HC_SCOPE_BASE + 32
#define HC_SCOPE_SAMPLE_MODE                            HC_SCOPE_BASE + 33
#define HC_SCOPE_SAMPLE_RATE                            HC_SCOPE_BASE + 34
#define HC_SCOPE_TYPE                                   HC_SCOPE_BASE + 35
#define HC_SCOPE_NUMBER_OF_POINTS_MIN                   HC_SCOPE_BASE + 36
#define HC_SCOPE_START_TIME                             HC_SCOPE_BASE + 37
#define HC_SCOPE_TIME_PER_RECORD                        HC_SCOPE_BASE + 38


//---------------------------------------------------------------------------
//   IIviScopeReferenceLevel Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_REFERENCE_LEVEL            HC_SCOPE_BASE + 39
#define HC_SCOPE_CONFIGURE_REFERENCE_LEVELS             HC_SCOPE_BASE + 40
#define HC_SCOPE_REFERENCE_HIGH                         HC_SCOPE_BASE + 41
#define HC_SCOPE_REFERENCE_LOW                          HC_SCOPE_BASE + 42
#define HC_SCOPE_REFERENCE_MID                          HC_SCOPE_BASE + 43


//---------------------------------------------------------------------------
//   IIviScopeChannels Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_CHANNELS                   HC_SCOPE_BASE + 44
#define HC_SCOPE_CHANNELS_ITEM                          HC_SCOPE_BASE + 45
#define HC_SCOPE_CHANNELS_COUNT                         HC_SCOPE_BASE + 46
#define HC_SCOPE_CHANNELS_NAME                          HC_SCOPE_BASE + 47


//---------------------------------------------------------------------------
//   IIviScopeChannel Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_CHANNEL                    HC_SCOPE_BASE + 48
#define HC_SCOPE_CONFIGURE_CHANNEL                      HC_SCOPE_BASE + 49
#define HC_SCOPE_CONFIGURE_CHARACTERISTICS              HC_SCOPE_BASE + 50
#define HC_SCOPE_PROBE_SENSE                            HC_SCOPE_BASE + 51
#define HC_SCOPE_COUPLING                               HC_SCOPE_BASE + 52
#define HC_SCOPE_ENABLED                                HC_SCOPE_BASE + 53
#define HC_SCOPE_INPUT_FREQUENCY_MAX                    HC_SCOPE_BASE + 54
#define HC_SCOPE_INPUT_IMPEDANCE                        HC_SCOPE_BASE + 55
#define HC_SCOPE_OFFSET                                 HC_SCOPE_BASE + 56
#define HC_SCOPE_PROBE_ATTENUATION                      HC_SCOPE_BASE + 57
#define HC_SCOPE_RANGE                                  HC_SCOPE_BASE + 58


//---------------------------------------------------------------------------
//   IIviScopeMeasurements Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_MEASUREMENTS               HC_SCOPE_BASE + 59
#define HC_SCOPE_MEASUREMENTS_ITEM                      HC_SCOPE_BASE + 60
#define HC_SCOPE_MEASUREMENTS_COUNT                     HC_SCOPE_BASE + 61
#define HC_SCOPE_MEASUREMENTS_NAME                      HC_SCOPE_BASE + 62
#define HC_SCOPE_INITIATE                               HC_SCOPE_BASE + 63
#define HC_SCOPE_ABORT                                  HC_SCOPE_BASE + 64
#define HC_SCOPE_AUTO_SETUP                             HC_SCOPE_BASE + 65
#define HC_SCOPE_STATUS                                 HC_SCOPE_BASE + 66
#define HC_SCOPE_IS_WAVEFORM_ELEMENT_INVALID            HC_SCOPE_BASE + 67
```

```
//---------------------------------------------------------------------------
//   IIviScopeMeasurement Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_MEASUREMENT             HC_SCOPE_BASE + 68
#define HC_SCOPE_FETCH_WAVEFORM                      HC_SCOPE_BASE + 69
#define HC_SCOPE_FETCH_WAVEFORM_MEASUREMENT          HC_SCOPE_BASE + 70
#define HC_SCOPE_FETCH_WAVEFORM_MIN_MAX              HC_SCOPE_BASE + 71
#define HC_SCOPE_READ_WAVEFORM                       HC_SCOPE_BASE + 72
#define HC_SCOPE_READ_WAVEFORM_MEASUREMENT           HC_SCOPE_BASE + 73
#define HC_SCOPE_READ_WAVEFORM_MIN_MAX               HC_SCOPE_BASE + 74


//---------------------------------------------------------------------------
//   IIviScopeTrigger Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_TRIGGER                 HC_SCOPE_BASE + 75
#define HC_SCOPE_CONFIGURE                           HC_SCOPE_BASE + 76
#define HC_SCOPE_CONTINUOUS                          HC_SCOPE_BASE + 77
#define HC_SCOPE_TRIGGER_COUPLING                    HC_SCOPE_BASE + 78
#define HC_SCOPE_LEVEL                               HC_SCOPE_BASE + 79
#define HC_SCOPE_MODIFIER                            HC_SCOPE_BASE + 80
#define HC_SCOPE_SOURCE                              HC_SCOPE_BASE + 81
#define HC_SCOPE_HOLDOFF                             HC_SCOPE_BASE + 82
#define HC_SCOPE_TRIGGER_TYPE                        HC_SCOPE_BASE + 83
#define HC_SCOPE_TRIGGER_AC_LINE                     HC_SCOPE_BASE + 84
#define HC_SCOPE_TRIGGER_EDGE                        HC_SCOPE_BASE + 85
#define HC_SCOPE_TRIGGER_GLITCH                      HC_SCOPE_BASE + 86
#define HC_SCOPE_TRIGGER_RUNT                        HC_SCOPE_BASE + 87
#define HC_SCOPE_TRIGGER_TV                          HC_SCOPE_BASE + 88
#define HC_SCOPE_TRIGGER_WIDTH                       HC_SCOPE_BASE + 89


//---------------------------------------------------------------------------
//   IIviScopeAcLine Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_TRIGGER_AC_LINE         HC_SCOPE_BASE + 90
#define HC_SCOPE_AC_LINE_SLOPE                       HC_SCOPE_BASE + 91


//---------------------------------------------------------------------------
//   IIviScopeEdge Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_TRIGGER_EDGE            HC_SCOPE_BASE + 92
#define HC_SCOPE_CONFIGURE_EDGE_SOURCE               HC_SCOPE_BASE + 93
#define HC_SCOPE_SLOPE                               HC_SCOPE_BASE + 94


//---------------------------------------------------------------------------
//   IIviScopeGlitch Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_TRIGGER_GLITCH          HC_SCOPE_BASE + 95
#define HC_SCOPE_GLITCH_CONFIGURE_TRIGGER            HC_SCOPE_BASE + 96
#define HC_SCOPE_GLITCH_CONDITION                    HC_SCOPE_BASE + 97
#define HC_SCOPE_GLITCH_POLARITY                     HC_SCOPE_BASE + 98
#define HC_SCOPE_GLITCH_WIDTH                        HC_SCOPE_BASE + 99
```

```
//---------------------------------------------------------------------------
//   IIviScopeRunt Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_TRIGGER_RUNT            HC_SCOPE_BASE + 100
#define HC_SCOPE_RUNT_CONFIGURE_TRIGGER              HC_SCOPE_BASE + 101
#define HC_SCOPE_RUNT_POLARITY                       HC_SCOPE_BASE + 102
#define HC_SCOPE_RUNT_THRESHOLD_HIGH                 HC_SCOPE_BASE + 103
#define HC_SCOPE_RUNT_THRESHOLD_LOW                  HC_SCOPE_BASE + 104


//---------------------------------------------------------------------------
//   IIviScopeTV Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_TRIGGER_TV              HC_SCOPE_BASE + 105
#define HC_SCOPE_TV_CONFIGURE_TRIGGER                HC_SCOPE_BASE + 106
#define HC_SCOPE_TV_LINE_NUMBER                      HC_SCOPE_BASE + 107
#define HC_SCOPE_TV_EVENT                            HC_SCOPE_BASE + 108
#define HC_SCOPE_TV_POLARITY                         HC_SCOPE_BASE + 109
#define HC_SCOPE_TV_SIGNAL_FORMAT                    HC_SCOPE_BASE + 110


//---------------------------------------------------------------------------
//   IIviScopeWidth Interface
//---------------------------------------------------------------------------

#define HC_SCOPE_I_IVI_SCOPE_TRIGGER_WIDTH           HC_SCOPE_BASE + 111
#define HC_SCOPE_WIDTH_CONFIGURE_TRIGGER             HC_SCOPE_BASE + 112
#define HC_SCOPE_WIDTH_CONDITION                     HC_SCOPE_BASE + 113
#define HC_SCOPE_WIDTH_POLARITY                      HC_SCOPE_BASE + 114
#define HC_SCOPE_WIDTH_THRESHOLD_LOW                 HC_SCOPE_BASE + 115
#define HC_SCOPE_WIDTH_THRESHOLD_HIGH                HC_SCOPE_BASE + 116


//---------------------------------------------------------------------------
//   Help Strings
//---------------------------------------------------------------------------

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

#define HS_SCOPE_HRESULTS \
"IVI Scope class defined HRESULTS"

#define HS_SCOPE_AC_LINE_SLOPE_ENUM \
"IVI Scope class-compliant values for AC line trigger Slope"

#define HS_SCOPE_ACQUISITION_STATUS_ENUM \
"IVI Scope class-compliant values for the status parameter of the \
acquisition Status method"

#define HS_SCOPE_ACQUISITION_TYPE_ENUM \
"IVI Scope class-compliant values for Acquisition Type"

#define HS_SCOPE_GLITCH_CONDITION_ENUM \
"IVI Scope class-compliant values for glitch trigger Condition"

#define HS_SCOPE_GLITCH_POLARITY_ENUM \
"IVI Scope class-compliant values for glitch trigger Polarity"

#define HS_SCOPE_INTERPOLATION_ENUM \
```

```
"IVI Scope class-compliant values for acquisition Interpolation"

#define HS_SCOPE_MEASUREMENT_ENUM \
"IVI Scope class-compliant values for the MeasFunction parameter of the \
measurement Read and Fetch methods"

#define HS_SCOPE_RUNT_POLARITY_ENUM \
"IVI Scope class-compliant values for runt trigger Polarity"

#define HS_SCOPE_SAMPLE_MODE_ENUM \
"IVI Scope class-compliant values for acquisition SampleMode"

#define HS_SCOPE_TIME_OUT_ENUM \
"IVI Scope class-compliant values for MaxTimeMilliseconds parameter of the
measurement \
Read and Fetch methods"

#define HS_SCOPE_TRIGGER_COUPLING_ENUM \
"IVI Scope class-compliant values for trigger Coupling"

#define HS_SCOPE_TRIGGER_MODIFIER_ENUM \
"IVI Scope class-compliant values for trigger Modifier"

#define HS_SCOPE_TRIGGER_SLOPE_ENUM \
"IVI Scope class-compliant values for edge trigger Slope"

#define HS_SCOPE_TRIGGER_TYPE_ENUM \
"IVI Scope class-compliant values for trigger Type"

#define HS_SCOPE_TV_SIGNAL_FORMAT_ENUM \
"IVI Scope class-compliant values for TV trigger SignalFormat"

#define HS_SCOPE_TV_TRIGGER_EVENT_ENUM \
"IVI Scope class-compliant values for TV trigger Event"

#define HS_SCOPE_TV_TRIGGER_POLARITY_ENUM \
"IVI Scope class-compliant values for TV trigger Polarity"

#define HS_SCOPE_VERTICAL_COUPLING_ENUM \
"IVI Scope class-compliant values for channel Coupling"

#define HS_SCOPE_WIDTH_POLARITY_ENUM \
"IVI Scope class-compliant values for width trigger Polarity"

#define HS_SCOPE_WIDTH_CONDITION_ENUM \
"IVI Scope class-compliant values for width trigger Condition"


//--------------------------------------------------------------------------
//   IVI Custom Driver Root Level Interface
//--------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE \
"IVI Scope class-compliant root interface"

#define HS_SCOPE_ACQUISITION \
"Pointer to the class-compliant IIviScopeAcquisition interface"

#define HS_SCOPE_CHANNELS \
"Pointer to the class-compliant IIviScopeChannels interface"

#define HS_SCOPE_MEASUREMENTS \
"Pointer to the class-compliant IIviScopeMeasurements interface"
```

```
#define HS_SCOPE_REFERENCE_LEVEL \
"Pointer to the class-compliant IIviScopeReferenceLevel interface"

#define HS_SCOPE_TRIGGER \
"Pointer to the class-compliant IIviScopeTrigger interface"


//--------------------------------------------------------------------------
//   IIviScopeAcquisition Interface
//--------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_ACQUISITION \
"IVI Scope class-compliant acquisition interface"

#define HS_SCOPE_CONFIGURE_RECORD \
"Configures the most commonly used attributes of the oscilloscope's \
acquisition subsystem: time per record, minimum record length, and the \
acquisition start time."

#define HS_SCOPE_AVERAGES \
"The number of waveforms the oscilloscope acquires and averages before \
returning to the idle state."

#define HS_SCOPE_ENVELOPES \
"The number of waveforms the oscilloscope acquires and analyzes to create the \
minimum and maximum waveforms, before returning to the idle state.  Applies \
only when acquisition Type is Envelope"

#define HS_SCOPE_INTERPOLATION \
"The interpolation method the oscilloscope uses when it cannot \
sample a voltage for every point in the waveform record."

#define HS_SCOPE_RECORD_LENGTH \
"The actual number of points the oscilloscope acquires for each \
channel.  It is equal to or greater than the minimum number of \
points specified with the Horizontal Minimun Number of Points property."

#define HS_SCOPE_SAMPLE_MODE \
"The sample mode the oscilloscope is currently using."

#define HS_SCOPE_SAMPLE_RATE \
"The effective digitizing rate using the current configuration.  The \
units are samples per second."

#define HS_SCOPE_TYPE \
"How the oscilloscope acquires data and fills the waveform record.  \
When set to Envelope or Peak Detect, the oscilloscope acquires minimum and \
maximum waveforms."

#define HS_SCOPE_NUMBER_OF_POINTS_MIN \
"The minimum number of points which can be in a waveform record for each \
channel.  It configures the record length that the oscilloscope uses for \
waveform acquisition.  The Record Length property returns the actual record \
length."

#define HS_SCOPE_START_TIME \
"The length of time from the trigger event to the first point in the waveform \
record.  The units are seconds.  If positive, the first point in the waveform \
occurs after the trigger.  If negative, the first point in the waveform \
occurs before the trigger."

#define HS_SCOPE_TIME_PER_RECORD \
```

```
"The time in seconds that corresponds to the record length."


//--------------------------------------------------------------------------------
//   IIviScopeReferenceLevel Interface
//--------------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_REFERENCE_LEVEL \
"IVI Scope class-compliant reference level interface"

#define HS_SCOPE_CONFIGURE_REFERENCE_LEVELS \
"Configures the reference levels for waveform measurements, low, mid, and
high."

#define HS_SCOPE_REFERENCE_HIGH \
"The high reference for waveform measurements.  It is a percentage of \
the difference between the Voltage High and Voltage Low.  Voltage High \
and Voltage Low may be calculated using either the min/max or histogram
methods."

#define HS_SCOPE_REFERENCE_LOW \
"The low reference for waveform measurements.  It is a percentage of \
the difference between the Voltage High and Voltage Low.  Voltage High \
and Voltage Low may be calculated using either the min/max or histogram
methods."

#define HS_SCOPE_REFERENCE_MID \
"The mid reference for waveform measurements.  It is a percentage of \
the difference between the Voltage High and Voltage Low.  Voltage High \
and Voltage Low may be calculated using either the min/max or histogram
methods."


//--------------------------------------------------------------------------------
//   IIviScopeChannels Interface
//--------------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_CHANNELS \
"IVI Scope class-compliant channel collection interface"

#define HS_SCOPE_CHANNELS_ITEM \
"An interface reference pointer to one of the IviScopeChannel interfaces \
which is selected by the channel name."

#define HS_SCOPE_CHANNELS_COUNT \
"The number of channels."

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


//--------------------------------------------------------------------------------
//   IIviScopeChannel Interface
//--------------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_CHANNEL \
"IVI Scope class-compliant channel interface"

#define HS_SCOPE_CONFIGURE_CHANNEL \
"Configures the most commonly used properties of the oscilloscope channel \
sub-system.  They are the range, offset, coupling, probe attenuation, and \
whether the channel is enabled."
```

```
#define HS_SCOPE_CONFIGURE_CHARACTERISTICS \
"Configures the properties that control the electrical characteristics of \
the channel.  They are the input impedance and the maximum frequency of the \
input signal."

#define HS_SCOPE_PROBE_SENSE \
"The probe attenuation value the oscilloscope automatically senses.  If the \
automatic probe sense is disabled, its value is the manual probe attenuation \
setting."

#define HS_SCOPE_COUPLING \
"How the oscilloscope couples the input signal."

#define HS_SCOPE_ENABLED \
"If True, the oscilloscope acquires a waveform for this channel when the \
Initiate Acquisition, Read Waveform, Read Min Max Waveform, or Read Waveform \
Measurement methods are called."

#define HS_SCOPE_INPUT_FREQUENCY_MAX \
"The maximum input frequency of this channel.  It the frequency at which the \
input circuitry attenuates the input signal by 3 dB.  The units are hertz."

#define HS_SCOPE_INPUT_IMPEDANCE \
"The input impedance of this channel.  The units are ohms."

#define HS_SCOPE_OFFSET \
"The location of the center of the range that you specify with the Range \
attribute.  The units are volts, with respect to ground.  For example, to \
acquire a sine wave spanning 0.0 to 10.0 volts, set Offset to 5.0 volts."

#define HS_SCOPE_PROBE_ATTENUATION \
"The scaling factor by which the probe attenuates the input signal.  \
For example, with a 10:1 probe, the value is 10.0."

#define HS_SCOPE_RANGE \
"The absolute value of the input range the oscilloscope can acquire for the \
channel.  The units are volts.  For example, to acquire a sine wave spanning \
-5.0 to 5.0 volts, set Range to 10.0 volts."



//--------------------------------------------------------------------------
//   IIviScopeMeasurements Interface
//--------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_MEASUREMENTS \
"IVI Scope class-compliant measurement collection interface"

#define HS_SCOPE_MEASUREMENTS_ITEM \
"An interface reference pointer to one of the IviScopeMeasurement interfaces \
which is selected by the channel name."

#define HS_SCOPE_MEASUREMENTS_COUNT \
"The number of measurements."

#define HS_SCOPE_MEASUREMENTS_NAME \
"The channel name for given index."

#define HS_SCOPE_INITIATE \
"Initiates a waveform acquisition.  The oscilloscope leaves the Idle state \
and waits for a trigger.  The oscilloscope acquires a waveform for each \
enabled channel."
```

```
#define HS_SCOPE_ABORT \
"Aborts an acquisition and returns the oscilloscope to the Idle state."

#define HS_SCOPE_AUTO_SETUP \
"Automatically configures all the oscilloscopes settings based on the \
input signals."

#define HS_SCOPE_STATUS \
"Returns whether an acquisition is in progress, complete, or if the status \
is unknown."

#define HS_SCOPE_IS_WAVEFORM_ELEMENT_INVALID \
"Returns False if an element in a waveform array returned by the driver \
contains a valid voltage.  Returns True if an element in a waveform array \
returned by the driver contains a value indicating that the oscilloscope \
could not sample a voltage."


//--------------------------------------------------------------------------
//   IIviScopeMeasurement Interface
//--------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_MEASUREMENT \
"IVI Scope class-compliant measurement interface"

#define HS_SCOPE_FETCH_WAVEFORM \
"Returns a previously acquired waveform for this channel. The acquisition \
must be made prior to calling this method.  Call this function separately \
for each channel."

#define HS_SCOPE_FETCH_WAVEFORM_MEASUREMENT \
"Returns a previously acquired waveform measurement for this channel.  The \
acquisition must be made prior to calling this method.  Call this \
function separately for each measurement."

#define HS_SCOPE_FETCH_WAVEFORM_MIN_MAX \
"Returns the previously acquired minimum and maximum waveforms for this \
specified channel.  The acquisition must be made prior to calling this method.
\
Call this function separately for each channel."

#define HS_SCOPE_READ_WAVEFORM \
"Initiates an acquisition on all enabled channels, waits (up to
MaxTimeMilliseconds) \
for the acquisition to complete, and returns the waveform for this \
channel.  Call FetchWaveform to obtain the waveforms for other channels."

#define HS_SCOPE_READ_WAVEFORM_MEASUREMENT \
"Initiates an acquisition on all enabled channels, waits (up to
MaxTimeMilliseconds) \
for the acquisition to complete, and returns the measurement for this \
channel.  Call FetchWaveformMeasurement to obtain other measurements for \
this or other channels."

#define HS_SCOPE_READ_WAVEFORM_MIN_MAX \
"Initiates an acquisition on all enabled channels, waits (up to
MaxTimeMilliseconds) \
for the acquisition to complete, and returns the min/max waveforms for this \
channel.  Call FetchMinMaxWaveform to obtain the min/max waveforms \
for other channels."
```

```
//----------------------------------------------------------------------------
//   IIviScopeTrigger Interface
//----------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_TRIGGER \
"IVI Scope class-compliant trigger interface"

#define HS_SCOPE_CONFIGURE \
"Configures trigger Type and Holdoff.  Holdoff units are seconds."

#define HS_SCOPE_CONTINUOUS \
"If True, the oscilloscope waits trigger holdoff seconds after a \
waveform acquisition is complete and then immediatley enters the wait \
for trigger state without passing through the idle state."

#define HS_SCOPE_TRIGGER_COUPLING \
"How the oscilloscope couples the trigger source."

#define HS_SCOPE_LEVEL \
"The voltage threshold for the trigger subsystem.  The units are volts."

#define HS_SCOPE_MODIFIER \
"The trigger modifier determines the oscilloscope's behavior in the absence \
of a trigger."

#define HS_SCOPE_SOURCE \
"The signal the oscilloscope monitors for a trigger. It can be channel or \
one of many other values."

#define HS_SCOPE_HOLDOFF \
"The length of time the oscilloscope waits after it fills the acquisition \
buffer until the oscilloscope enables the trigger subsystem to detect \
another trigger.  The units are seconds."

#define HS_SCOPE_TRIGGER_TYPE \
"The kind of event that triggers the osciolloscope."

#define HS_SCOPE_TRIGGER_AC_LINE \
"Pointer to the class-compliant IIviScopeTriggerAcLine interface"

#define HS_SCOPE_TRIGGER_EDGE \
"Pointer to the class-compliant IIviScopeTriggerEdge interface"

#define HS_SCOPE_TRIGGER_GLITCH \
"Pointer to the class-compliant IIviScopeTriggerGlitch interface"

#define HS_SCOPE_TRIGGER_RUNT \
"Pointer to the class-compliant IIviScopeTriggerRunt interface"

#define HS_SCOPE_TRIGGER_TV \
"Pointer to the class-compliant IIviScopeTriggerTv interface"

#define HS_SCOPE_TRIGGER_WIDTH \
"Pointer to the class-compliant IIviScopeTriggerWidth interface"


//----------------------------------------------------------------------------
//   IIviScopeAcLine Interface
//----------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_TRIGGER_AC_LINE \
"IVI Scope class-compliant AC line trigger interface"
```

```
#define HS_SCOPE_AC_LINE_SLOPE \
"The slope of the zero crossing upon which the scope triggers."


//------------------------------------------------------------------------------
//   IIviScopeEdge Interface
//------------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_TRIGGER_EDGE \
"IVI Scope class-compliant edge trigger interface"

#define HS_SCOPE_CONFIGURE_EDGE_SOURCE \
"Configures the conidtions for edge trigger.  An edge trigger occurs when \
the trigger source signal passes through the trigger level with the specified \
slope."

#define HS_SCOPE_SLOPE \
"The slope, a rising or a falling edge, that triggers the oscilloscope."



//------------------------------------------------------------------------------
//   IIviScopeGlitch Interface
//------------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_TRIGGER_GLITCH \
"IVI Scope class-compliant glitch trigger interface"

#define HS_SCOPE_GLITCH_CONFIGURE_TRIGGER \
"Configure the glitch trigger Source, Level, Width, Polarity, and Condition.  \
A glitch trigger occurs when the edge of a pulse that matches the Width and \
Polarity crosses the specified Level (in Volts)."

#define HS_SCOPE_GLITCH_CONDITION \
"The glitch condition detemermines whether the oscilloscope triggers on a \
a pulse with a width less than or greater than the glitch width value."

#define HS_SCOPE_GLITCH_POLARITY \
"The polarity of the glitch that triggers the oscilloscope."

#define HS_SCOPE_GLITCH_WIDTH \
"The glitch width.  The units are seconds."



//------------------------------------------------------------------------------
//   IIviScopeRunt Interface
//------------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_TRIGGER_RUNT \
"IVI Scope class-compliant runt trigger interface"

#define HS_SCOPE_RUNT_CONFIGURE_TRIGGER \
"Configures the runt trigger Source, ThresholdLow, ThresholdHigh, and \
Polarity.  A runt trigger occurs when the trigger signal crosses one of the \
runt thresholds twice without crossing the other runt threshold."

#define HS_SCOPE_RUNT_POLARITY \
"The polarity of the runt that triggers the oscilloscope."

#define HS_SCOPE_RUNT_THRESHOLD_HIGH \
"The high threshold the oscilloscope uses for runt triggering.  \
The units are volts."
```

```
#define HS_SCOPE_RUNT_THRESHOLD_LOW \
"The low threshold the oscilloscope uses for runt triggering.  \
The units are volts."



//---------------------------------------------------------------------------
//   IIviScopeTV Interface
//---------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_TRIGGER_TV \
"IVI Scope class-compliant TV trigger interface"

#define HS_SCOPE_TV_CONFIGURE_TRIGGER \
"Configures the TV trigger source, signal format, event and polarity. "

#define HS_SCOPE_TV_LINE_NUMBER \
"The line on which the oscilloscope triggers.  The line number is absolute \
and not relative to the field of the TV signal."

#define HS_SCOPE_TV_EVENT \
"The event on which the oscilloscope triggers."

#define HS_SCOPE_TV_POLARITY \
"The polarity of the TV signal. "

#define HS_SCOPE_TV_SIGNAL_FORMAT \
"The format of the TV signal on which the oscilloscope triggers."



//---------------------------------------------------------------------------
//   IIviScopeWidth Interface
//---------------------------------------------------------------------------

#define HS_SCOPE_I_IVI_SCOPE_TRIGGER_WIDTH \
"IVI Scope class-compliant width trigger interface"

#define HS_SCOPE_WIDTH_CONFIGURE_TRIGGER \
"Configures the width trigger Source, Level, ThresholdLow, ThresholdHigh, \
Polarity, and Condition.  A width trigger occurs when a pulse, that passes \
through Level, with a width between or outside, the width thresholds is \
detected."

#define HS_SCOPE_WIDTH_CONDITION \
"The condition of a pulse that triggers the oscilloscope.  The condition is \
either inside or outside of the high and low thresholds."

#define HS_SCOPE_WIDTH_POLARITY \
"The polarity of the pulse that triggers the oscilloscope. "

#define HS_SCOPE_WIDTH_THRESHOLD_LOW \
"The low width threshold time.  The units are seconds. "

#define HS_SCOPE_WIDTH_THRESHOLD_HIGH \
"The high width threshold time.  The units are seconds. "

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