IVI-3.3: Standard Cross-Class Capabilities Specification

December 19, 2014 Edition
Revision 3.2

Important Information

This specification (IVI-3.3: Standard Cross-Class Capabilities Specification) 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`.

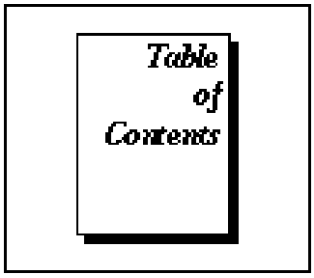
Warranty

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.



Standard Cross-Class Capabilities

IVI Standard Cross-Class Capabilities Revision History

This section is an overview of the revision history of the IVI-3.3 specification. Specific individual additions/modifications to the document in draft revisions are denoted with diff-marks, "|", in the right hand column of a line of text for which the change/modification applies.

Table 1-1. IVI Standard Cross-Class Capability Specification Revisions

Revision Number	Date of Revision	Revision Notes
Revision 1.0	April, 2002	First approved version. Voting Candidate 3 Approved, with minor edits agreed to by Working Group
Revision 1.1	June, 2006	Fix typographical errors in section 3.3.2.
Revision 1.1	March, 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 2.0	November 17, 2008	Add a list of standard interfaces / trigger sources with standard strings for each item, and standard identifier names for each item on the list.
Revision 3.0	June 9, 2010	Incorporated .NET
Revision 3.1	October 14, 2010	Clarify the legal use of the Get<RC>Name method.
Revision 3.1	April 15, 2011	Editorial change - format Section 4.4.1 correctly.
Revision 3.1	June 30, 2011	Editorial change - add a reference to the TriggerSource class.
Revision 3.2	March 10, 2012	Editorial Changes - Add three trigger strings to section 3.
Revision 3.2	December 19, 2014	Editorial Change – Removed a sentence in Section 4.5.1, Active<RC> Attribute.

API Versions

Architecture	Drivers that comply with version 3.1 comply with all of the versions below
C	1.0, 2.0, 3.0
COM	1.0, 2.0, 3.0

.NET	3.0

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

1. Overview of Standard Cross-Class Capabilities

The IVI-3.3 Standard Cross-Class Capabilities specification describes various capabilities which are common in at least two instrument classes.

An IVI class specification describes the attributes and functions required for a particular instrument class. Some attributes and functions should be defined identically in every instrument class. Those functions and attributes are described here to avoid gratuitous differences.. This document contains those functions and attributes that apply across multiple instrument classes.

1.1 Notation

<Class>	Designates the appropriate class prefix in mixed case form. For example, the function `<Class>_SendSoftwareTrigger` has the name `IviDMM_SendSWTrigger` in the IviDMM instrument class.
<CLASS>	Designates the appropriate class prefix in all upper case. For example, the attribute `<CLASS>_ATTR_SAMPLE_COUNT` has the name `IVIDMM_ATTR_SAMPLE_COUNT` in the IviDMM instrument class.

1.2 When to Define a Standard Cross-Class Capability

When in the course of developing a new class specification the working group identifies an instrument capability that it believes is common across multiple instrument classes, the working group chairperson should contact the chairperson of the Standard Cross-Class Capability working group and have the common capability entered into Standard Cross-Class Capabilities. This document then proceeds through the normal process of revising IVI documents.

1.3 When to Refer to a Standard Cross-Class Capability

Class specifications under development reference capabilities described in this document. The text in this document is not copied into the instrument class specification. If an existing instrument class specification undergoes a revision, the working group may choose to update the instrument class specification to refer to this document.

In some cases, class specific information may need to be added to an instrument class specification to further refine the behavior of a standard cross-class capability for that particular instrument class.

2. Software Triggering Capability

Description

This function always appears in a <Class>SoftwareTrigger Extension Group.

This function sends a software-generated trigger to the instrument. It is only applicable for instruments using interfaces or protocols which support an explicit trigger function. For example, with GPIB this function could send a group execute trigger to the instrument. Other implementations might send a *TRG command.

Since instruments interpret a software-generated trigger in a wide variety of ways, the precise response of the instrument to this trigger is not defined. Note that SCPI details a possible implementation.

This function should not use resources which are potentially shared by other devices (for example, the VXI trigger lines). Use of such shared resources may have undesirable effects on other devices.

This function should 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. To check the instrument status, call the appropriate error query function at the conclusion of the sequence.

The trigger source attribute must accept Software Trigger as a valid setting for this function to work. If the trigger source is not set to Software Trigger, this function does nothing and returns the error Trigger Not Software.

.NET Method Prototype

```
void SendSoftwareTrigger ();
```

COM Method Prototype

```
HRESULT SendSoftwareTrigger ();
```

C Function Prototype

```
ViStatus <Class>_SendSoftwareTrigger (ViSession Vi);
```

Parameters

Inputs	Description
Vi	Instrument handle

Return Values (C/COM)

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

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

Exception Class	Description
Trigger Not Software	The trigger source is not set to software trigger.

.NET Exceptions

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

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

Exception Class	Description
TriggerNotSoftwareException	The trigger source is not set to software trigger.

2.1 Software Trigger Error Completion Codes and Exception Class Definitions

Table 2-1 lists the error name as it is used throughout the instrument class specification and this document, a more complete description of the error, and the identifiers used in languages of interest to IVI with an associated value.

Table 2-1. Software Triggering Error and Completion Codes

Error Name	***Description***		
	Language	***Identifier***	***Value(hex)***
Trigger Not Software	The trigger source is not set to software trigger.		
	.NET	Ivi.Driver.TriggerNotSoftwareException	N/A
	C	<CLASS>_ERROR_TRIGGER_NOT_SOFTWARE	0xBFFA1001
	COM	E_<CLASS>_TRIGGER_NOT_SOFTWARE	0x80041001

Table 2-2 defines the format of the message string associated with the error. In C, this string is returned by the Error Message function. In COM, this string is the description contained in the ErrorInfo object. In .NET this string is the *Message* property of the exception class thrown by the method or property.

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

Table 2-2. Software Triggering Error Message Strings

Name	Message String
Trigger Not Software	"%s: Trigger source is not set to software trigger."

3. Standard Trigger Source Values

There are a variety of properties in the IVI APIs related to LXI arming and trigger sources (including Trigger Source, Advanced Destination, and Sample Trigger), where LXI LAN and trigger bus triggers apply. These values take four forms.

- In the IVI-C API, the value may take the form of a defined constant. For example, the DMM class specification defines a Trigger Source property. One of the allowed values of that property is `IVIDMM_VAL_TTL0`, defined to be 111.

- In the IVI-COM API, the value may take the form of an enumeration value. For example, the DMM class specification defines a Trigger Source enumeration, `IviDmmTriggerSourceEnum`. One of the allowed values of that enumeration is `IviDmmTriggerSourceTTL0`, defined to be 111.

- In the IVI.NET API, the value shall take the form of a string. In IVI-C and IVI-COM APIs, the value may take the form of a string. In all cases in IVI.NET class-compliant and instrument specific interfaces, such properties shall be typed as `string`.

- In the IVI.NET API, a static string value property from the `TriggerSource` class may be used. This is equivalent to using a standard string, since the properties of this class return only the standard strings listed in the table below. For a description of the `TriggerSource` class, refer to *IVI-3.18, .NET Utility Classes and Interfaces Specification*, Section 14, *Standard Trigger Source Values Class.*

All new IVI specifications approved after January 1, 2009, shall use strings for trigger sources and LXI arming sources. The strings shall be treated as case-insensitive. In cases where the value is not pre-defined by the driver, the driver shall preserve the case of the string that the user passed to the driver. In cases where the value is pre-defined by the driver, the driver shall return the driver-defined value.

The following table defines the string values that shall be used to represent LXI arming and trigger sources in instrument class APIs. Additional values may be defined if needed. Instrument class specifications need only reference values that are germane to properties or parameters in the class API.

Trigger Source	*Description*	*String Value*
None	No Interface (normally applies only to triggers)	"None", "", or String.Empty
Immediate	Trigger Immediately (normally applies only to triggers)	"Immediate"
External	External source, not an interface (normally applies only to triggers)	"External"
Internal	Internal source, not an interface (normally applies only to triggers)	"Internal"
Software	Software source, not an interface (normally applies only to triggers)	"Software"
GET	Group Execute Trigger	"GET"
ACLine	AC Line Interface	"ACLine"
Interval	Trigger at set intervals	"Interval"
LAN0	LAN0 (LXI defined "LAN0" LAN message)	"LAN0"
LAN1	LAN1 (LXI defined "LAN1" LAN message)	"LAN1"
LAN2	LAN2 (LXI defined "LAN2" LAN message)	"LAN2"
LAN3	LAN3 (LXI defined "LAN3" LAN message)	"LAN3"
LAN4	LAN4 (LXI defined "LAN4" LAN message)	"LAN4"
LAN5	LAN5 (LXI defined "LAN5" LAN message)	"LAN5"
LAN6	LAN6 (LXI defined "LAN6" LAN message)	"LAN6"
LAN7	LAN7 (LXI defined "LAN7" LAN message)	"LAN7"
LXI0	LXI Trigger Bus Line 0	"LXI0"
LXI1	LXI Trigger Bus Line 1	"LXI1"
LXI2	LXI Trigger Bus Line 2	"LXI2"
LXI3	LXI Trigger Bus Line 3	"LXI3"
LXI4	LXI Trigger Bus Line 4	"LXI4"
LXI5	LXI Trigger Bus Line 5	"LXI5"
LXI6	LXI Trigger Bus Line 6	"LXI6"
LXI7	LXI Trigger Bus Line 7	"LXI7"
TTL0	TTL Interface 0	"TTL0"
TTL1	TTL Interface 1	"TTL1"
TTL2	TTL Interface 2	"TTL2"
TTL3	TTL Interface 3	"TTL3"
TTL4	TTL Interface 4	"TTL4"
TTL5	TTL Interface 5	"TTL5"
TTL6	TTL Interface 6	"TTL6"
TTL7	TTL Interface 7	"TTL7"
ECL0	ECL Line 0	"ECL0"

ECL1	ECL Line 1	"ECL1"
PXI_CLK10	PXI 10MHz Clock Line	"PXI_CLK10"
PXI_STAR	PXI Star Interface	"PXI_STAR"
PXI_TRIG0	PXI Trigger Bus Line 0	"PXI_TRIG0"
PXI_TRIG1	PXI Trigger Bus Line 1	"PXI_TRIG1"
PXI_TRIG2	PXI Trigger Bus Line 2	"PXI_TRIG2"
PXI_TRIG3	PXI Trigger Bus Line 3	"PXI_TRIG3"
PXI_TRIG4	PXI Trigger Bus Line 4	"PXI_TRIG4"
PXI_TRIG5	PXI Trigger Bus Line 5	"PXI_TRIG5"
PXI_TRIG6	PXI Trigger Bus Line 6	"PXI_TRIG6"
PXI_TRIG7	PXI Trigger Bus Line 7	"PXI_TRIG7"
PXIe_CLK100	PXI Express 100MHz Clock Line	"PXIe_CLK100"
PXIe_DSTARA	PXI Express DStar Line A	"PXIe_DSTARA"
PXIe_DSTARB	PXI Express DStar Line B	"PXIe_DSTARB"
PXIe_DSTARC	PXI Express DStar Line C	"PXIe_DSTARC"
RTSI0	RTSI Bus Line 0	"RTSI0"
RTSI1	RTSI Bus Line 1	"RTSI1"
RTSI2	RTSI Bus Line 2	"RTSI2"
RTSI3	RTSI Bus Line 3	"RTSI3"
RTSI4	RTSI Bus Line 4	"RTSI4"
RTSI5	RTSI Bus Line 5	"RTSI5"
RTSI6	RTSI Bus Line 6	"RTSI6"
RTSI7	RTSI Bus Line 7	"RTSI7"

Note that several instrument class specifications were completed before this section was added, and some of them use values that differ from the above table. These specifications shall continue to use the IVI-C and IVI-COM values as originally defined, and existing class APIs will not be expected to conform to the above table. If these APIs are extended with new methods or properties, the new methods or properties shall use strings and shall conform to the above table. These specifications are:

- *IVI 4-1: IviScope Class Specification*
- *IVI 4-2: IviDmm Class Specification*
- *IVI 4-3: IviFgen Class Specification*
- *IVI 4-4: IviDCPwr Class Specificat*ion
- *IVI 4-6: IviSwtch Class Specificat*ion
- *IVI 4-7: IviPwrMeter Class Specificat*ion
- *IVI 4-8: IviSpecAn Class Specificat*ion
- *IVI 4-10: IviRFSigGen Class Specificat*ion

The IVI.NET APIs for these classes use strings consistently.

3.1 Attributes

Attributes that use the strings listed above shall include the following text in their description, where <source> stands for the attribute name:

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

4. Repeated Capability Group

4.1 Overview

Instrument classes often describe capabilities which can be repeated in a particular instrument. Some examples are channels, and traces. This section describes attributes and functions which an instrument class should use to provide.

Section 12, *Repeated Capabilities*, in *IVI 3.4: API Style Guide*, describes three styles for representing repeated capabilities. The following table (duplicated in IVI-3.4) shows the attributes and functions that are used to implement each of the styles in each of the supported IVI APIs.

Technique	API Type	IVI.NET	IVI-C	IVI-COM
Parameter Style	Attributes	<RC> Count	<RC> Count	<RC> Count <RC> Name (Index)
	Functions	Get <RC> Name	Get <RC> Name	
Collection Style	Attributes	<RC> Item <RC> Count <RC> Name		<RC> Item <RC> Count <RC> Name(Index)
Selector Style	Attributes	<RC> Count Active <RC>	<RC> Count Active <RC>	<RC> Count Active <RC> <RC> Name
	Functions	Get <RC> Name	Get <RC> Name Set Active <RC>	

These attributes and functions should be placed in the same capability group as the repeated capability with which they are associated.

This section uses the notation <RC> to indicate a repeated capability name. . The instrument class specification specifies the actual name of the repeated capability. For example, if the repeated capability is named Channel then <RC> Count would appear as Channel Count in the instrument class. The plural form is shown as <RC>s. While the plural for most English words is formed by adding an s, many exceptions exist. For example, the plural of Leaf is Leaves, the plural of Child is Children, and the plural of Capability is Capabilities. The instrument class specification uses the proper English plural; it does not blindly add an s. In cases where the repeated capability name should be lower case, <rc> is used.

4.1.1 Specifying Repeated Capability Attributes and Functions

Most repeated capabilities use the parameter style for IVI-C, and the collection style for IVI-COM and IVI.NET. This is the most common way to implement repeated capabilities in IVI class specifications. Repeated capabilities should be implemented this way unless there is a compelling reason not to.

A smaller number of repeated capabilities use the selector style for IVI-C, IVI-COM, and IVI.NET.[1]

One repeated capability, the IviFgen Channel, uses the parameter style for IVI-C, IVI-COM, and IVI.NET.

[1] Two repeated capabilities, the IviRFSigGenAnalog Modulation Source and the IviUpconverter Analog Modulation Source, use a limited variation that replaces the Active <RC> attribute and Set Active <RC> function with AM Source, FM Source, and PM Source attributes, which allow the client to select one or more sources as the modulating signal.

4.2 Attributes – Common

The following attribute is documented in the same way regardless of how repeated capabilities are specified:

- <RC> Count

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in the instrument class.

4.2.1 <RC> Count

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	RO	<RC>s	None	

.NET Property Name

<RC>s.Count

This property is inherited from IIviRepeatedCapabilityCollection.

COM Property Name

<RC>s.Count

C Constant Name

*PREFIX*_ATTR_<RC>_COUNT

Description

Specifies the number of <RC>s available.

.NET Exceptions

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

4.3 Attributes – Parameter / Collection Style Combination

The following attribute sections document how the <RC> Item and <RC> Name attributes should appear when a repeated capability is implemented using parameter style in IVI-C and collection style in IVI-COM and IVI.NET:

- <RC> Item (IVI-COM and IVI.NET Only)
- <RC> Name (COM only)

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in the instrument class.

4.3.1 <RC> Item (IVI-COM and IVI.NET Only)

Data Type	Access	Applies to	Coercion	High Level Functions
IIvi<class><RC>	RO	<RC>s	None	

.NET Property Name

I<ClassName><RC>[2] <RC>s[String name];

This indexer is inherited from IIviRepeatedCapabilityCollection. The name parameter uniquely identifies a particular <rc> in the <rc> collection.

COM Property Name

```
HRESULT <RC>s.Item([in]BSTR Name,
                   [out, retval] I<ClassName><RC> **pVal);
```

C Constant Name

N/A

Description

<RC> Item uniquely identifies an <rc> in the <rc>s collection. It returns an interface pointer which can be used to control the attributes and other functionality of that <rc>.

The Item property takes a <rc> name. If the user passes an invalid value for the Name parameter, the property returns an error.

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

.NET Exceptions

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

4.3.2 <RC> Name (IVI-COM and IVI.NET Only)

Data Type	Access	Applies to	Coercion	High Level Functions
ViString	R	<RC>	None	

.NET Property Name

<RC>s[].Name

This property is inherited from IIviRepeatedCapabilityIdentification.

[2] The name of the interface is controlled by the instrument class writers, but it should follow the form shown.

COM Property Name

```
HRESULT <RC>s.Name ([in] LONG Index,
                    [out, retval] BSTR* Name);
```

C Constant Name

`N/A`
(Use the Get <RC> Name function.)

Description

Returns the physical repeated capability identifier defined by the specific driver for the <rc> that corresponds to the index that the user specifies. If the driver defines a qualified <rc> name, this property returns the qualified name.

For C and COM, valid values for the `Index` parameter are between one and the value of the <RC> Count attribute, inclusive. If the user passes an invalid value for the `Index` parameter, the value of this attribute is an empty string.

.NET Exceptions

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

Compliance Notes

For an instrument with only one <RC>, i.e. the <RC> Count attribute is one, the driver may return an empty string.

4.4 Functions – Parameter / Collection Style Combination

The following function section documents how the Get <RC> Name function should appear when a repeated capability is implemented using parameter style in IVI-C and collection style in IVI-COM and IVI.NET:

- Get <RC> Name (IVI-C & IVI.NET Only)

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in the instrument class.

4.4.1 Get <RC> Name (IVI-C & IVI.NET Only)

Description

This function returns the specific driver defined <rc> name that corresponds to the one-based index that the user specifies. If the driver defines a qualified <rc> name, this function returns the qualified name. If the value that the user passes for the `Index` parameter is less than one or greater than the value of the <RC> Count attribute, the function returns an empty string in the `Name` parameter and returns an Invalid Value error.

Note that this method is defined for .NET for parameter style repeated capabilities. However, the Fgen class is the only class that uses parameter style repeated capabilities, and this approach is not recommended for future classes.

.NET Method Prototype

`String <RC>.GetName (Int32 index);`(For collection style repeated capabilities, use the <RC> Name property. This method is only defined for .NET for parameter style repeated capabilities. However, the Fgen class is the only class that uses parameter style repeated capabilities, and this approach is not recommended for future classes.)

COM Method Prototype

`N/A`

(Use the <RC> Name property.)

C Function Prototype

```
ViStatus _Prefix_Get<RC>Name (ViSession Vi,
                              ViInt32 Index,
                              ViInt32 NameBufferSize,
                              ViChar Name[]);
```

Parameters

Inputs	Description	Base Type
`Vi`	Unique identifier for an IVI session	`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 `<RC>Name` parameter.	`ViInt32`

Outputs	Description	Base Type
`Name`	The buffer into which the function returns the name that corresponds to the index the user specifies. The caller may pass `VI_NULL` for this parameter if the `NameBufferSize` parameter is 0.	`ViChar[]`

Return Values (C)

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

Compliance Notes

1. For an instrument with only one <RC>, that is the <RC> Count attribute is one, the driver may return an empty string.

2. Refer to Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters*, in *IVI-3.2 Inherent Capabilities Specification* for rules regarding the `NameBufferSize` and `Name` parameters.

4.5 Attributes –Selector Style

The following attribute sections document how the Active <RC> and <RC> Name attributes should appear when a repeated capability is implemented using the selector style in IVI-C, IVI-COM, and IVI.NET:

- Active <RC>
- <RC> Name (IVI-COM Only)

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in the instrument class.

4.5.1 Active <RC>

Data Type	Access	Applies to	Coercion	High Level Functions
ViString	RW	<RC>	None	

COM Property Name

<RC>.Active<RC>

COM Property Name

<RC>.Active<RC>

C Constant Name

*PREFIX*_ATTR_ACTIVE_<RC>

Description

Specifies the <rc> which is currently active.

.NET Exceptions

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

4.5.2 <RC> Name (IVI-COM Only)

Data Type	Access	Applies to	Coercion	High Level Functions
ViString	R	<RC>	None	

.NET Property Name

N/A
(Use the Get <RC> Name function.)

COM Property Name

```
HRESULT <RC>.Name ([in] LONG Index,
                   [out, retval] BSTR* Name);
```

C Constant Name

N/A
(Use the Get <RC> Name function.)

Description

Returns the physical repeated capability identifier defined by the specific driver for the <rc> that corresponds to the index that the user specifies. If the driver defines a qualified <rc> name, this property returns the qualified name.

Valid values for the Index parameter are between one and the value of the <RC> Count attribute, inclusive. If the user passes an invalid value for the Index parameter, the value of this attribute is an empty string.

Compliance Notes

For an instrument with only one <RC>, i.e. the <RC> Count attribute is one, the driver may return an empty string.

4.6 Functions –Selector Style

The following function sections documents how the Get <RC> Name and Set Active <RC> functions should appear when a repeated capability is implemented using selector style in IVI-C, IVI-COM, and IVI.NET:

- Get <RC> Name (IVI-C and IVI.NET Only)
- Set Active <RC> (IVI-Cand IVI-COM Only)

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in the instrument class.

4.6.1 Get <RC> Name (IVI-C and IVI.NET Only)

Description

This function returns the specific driver defined <rc> name that corresponds to the one-based index that the user specifies. If the driver defines a qualified <rc> name, this function returns the qualified name. If the value that the user passes for the `Index` parameter is less than one or greater than the value of the <RC> Count attribute, the function returns an empty string in the `Name` parameter and returns an Invalid Value error.

.NET Method Prototype

```
String <RC>.GetName (Int32 index);
```

COM Method Prototype

`N/A`

(Use the <RC> Name property.)

C Function Prototype

```
ViStatus _Prefix_Get<RC>Name (ViSession Vi,
                              ViInt32 Index,
                              ViInt32 NameBufferSize,
                              ViChar Name[]);
```

Parameters

Inputs	Description	Base Type
`Vi`	Unique identifier for an IVI session	`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 `<RC>Name` parameter.	`ViInt32`

Outputs	Description	Base Type
`Name (C)`	The buffer into which the function returns the name that corresponds to the index the user specifies. The caller may pass `VI_NULL` for this parameter if the `NameBufferSize` parameter is 0.	`ViChar[]`
`Return value (.NET)`	The name that corresponds to the index the user specified.	`ViChar[]`

Return Values (C)

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

.NET Exceptions

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

Compliance Notes

1. For an instrument with only one <RC>, that is the <RC> Count attribute is one, the driver may return an empty string.

2. Refer to Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters*, in *IVI-3.2 Inherent Capabilities Specification* for rules regarding the `NameBufferSize` and `Name` parameters.

4.6.2 Set Active <RC> (IVI-C Only)

Description

This function selects one of the available <rc>s, and makes it the active <rc>.

.NET Method Prototype

```
N/A
```

(Use the Active Marker property.)

COM Method Prototype

```
N/A
```

(Use the Active Marker property.)

C Function Prototype

```
ViStatus Prefix_SetActive<RC> (ViSession Vi,
                               ViConstString Name);
```

Parameters

Inputs	Description	Base Type
`Vi`	Unique identifier for an IVI session	`ViSession`
`Name`	A string specifying a particular capability.	`ViString`

Return Values (C)

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

5. Automatic Setting Attributes

5.1 Overview

Instruments and drivers contain algorithms which automatically adjust settings based on other settings or characteristics of input signals. This specification does not specify when the automatic algorithm is run. For example, it may be run immediately, or later at a trigger event or measurement. Typically, these algorithms can be enabled and disabled.

Section 7, *Controlling Automatic Setting Attributes*, in *IVI 3.4: API Style Guide*, describes the attributes and methods that are needed to control automatic settings, and define some of the terms used in this section.

In the following sections, <Name> represents the name primary attribute. In most cases, the name of the automatic setting attribute will be "<Name> Auto"[3], though in some cases "Auto <Name>" is allowed. "<Name> Auto" will be used in the following sections. Note that <Intf> is that name of the interface reference property or properties used to access the attributes and <Type> is the data type of the primary attribute.

Many automatic setting attributes are defined as Booleans, and allow only for turning the automatic setting "On" or "Off". In some cases the automatic setting attribute also supports a "Once" setting, which requires a specialized Auto enumeration to be defined in the instrument class, which allows for values of "Off" (=0), "On" (=1), and "Once" (=2).

[3] If Auto <Name> is a recognized domain convention (for example, Auto Range or Auto Zero), then Auto may be placed before the name.

5.2 Attributes

The following attribute sections document how the <Name> and <Name> Auto attributes should appear when an automatic setting attribute is implemented in IVI-C, IVI-COM, and IVI.NET: The first example of the automatic setting attribute (<Name> Auto) is a 2-state Boolean, and the second example is a 3-state enumeration. Note that values for Data Type, Applies To, Coercion, High Level Functions, and Description are example values only, and may vary between attributes.

- <Name>
- <Name> Auto

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in the instrument class.

5.2.1 <Name>

Data Type	Access	Applies to	Coercion	High Level Functions
<Type>	R/W	N/A	N/A	Configure Level

.NET Property Name

<Intf>.<Name>

COM Property Name

<Intf>.<Name>

C Constant Name

IVISPECAN_ATTR_<Name>

Description

[Put property description here.]

The act of setting the <Name> property shall have the side effect of setting the <Name> Auto attribute to False.

.NET Exceptions

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

5.2.2 <Name> Auto (Defined as Boolean)

Data Type	Access	Applies to	Coercion	High Level Functions
ViBoolean	R/W	N/A	N/A	Configure

.NET Property Name

<Intf>.<Name>Auto

COM Property Name

<Intf>.<Name>Auto

C Constant Name

<CLASSNAME>_ATTR_<NAME>AUTO

Description

[Adapt the following text for the actual property.]

If set to True, <Name> is automatically determined. If set to False, <Name> shall go to the manual setting mode and retain the current value for <Name>.

When <Name> Auto is True, the actual value for <name> as automatically determined by the instrument is returned by the <Name> attribute.

.NET Exceptions

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

5.2.3 <Name> Auto (Defined as an Auto enumeration with a “Once” value)

Data Type	Access	Applies to	Coercion	High Level Functions
ViInt32	R/W	N/A	Up	Configure Level

.NET Property Name

<Intf>.<Name>Auto

.NET Enumeration Name

Auto

COM Property Name

<Intf>.<Name>Auto

COM Enumeration Name

<ClassName><Name>AutoEnum

C Constant Name

<CLASSNAME>_ATTR_<NAME>AUTO

Description

[Adapt the following text for the actual property.]

Specifies if the instrument sets <Name> automatically, as described in the following table.

Setting the <Name> attribute also sets the <Name> Auto attribute to Auto Off.

Name	***Description***		
		Language	***Identifier***
Auto On	Sets the instrument to determine <Name> automatically. When On, the actual value for <name> as automatically determined by the instrument is returned by the <Name> attribute		
		.NET	Auto.On
		C	<CLASSNAME>_VAL_<NAME>AUTO_ON
		COM	<ClassName><Name>AutoOn
Auto Off	Disables auto-<Name>. The instrument sets the <Name> attribute to the value it most recently determined.		
		.NET	Auto.Off
		C	<CLASSNAME>_VAL_<NAME>AUTO_OFF

		COM	<ClassName><Name>AutoOff
Auto Once	Sets the instrument to calculate <Name> exactly once, before its next use. After its next use, the driver uses the instrument-determined value of <Name> for subsequent uses.		
		.NET	Auto.Once
		C	<CLASSNAME>_VAL_<NAME>AUTO_ONCE
		COM	<ClassName><Name>AutoOnce

.NET Exceptions

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

5.3 Functions

The following function section documents how the <Name> and <Name> Auto attributes should appear when used as parameters in a function or method. Note that this section is provided as an example – the function name and other parameters, as well as description and parameter descriptions and compliance notes will all vary according to the actual function. Also, only the Boolean (2-state) case for <Name> Auto is shown. The auto enumeration case (3-state) is similar.

- Configure <Name>

This section describes the behavior and requirements of each function.

5.3.1 Configure <Name>

Description

[Adapt the description to the actual method.]

.NET Method Prototype

```
void <Intf>.Configure (<Type> <name>);

void <Intf>.Configure (Boolean <name>Auto);
```

COM Method Prototype

```
HRESULT <Intf>.Configure ([in] VARIANT_BOOL <Name>Auto,
                          [in] <Type> <Name>);
```

C Function Prototype

```
ViStatus <ClassName>_Configure (ViSession Vi,
                                ViBoolean <Name>Auto,
                                <Type> <Name>);
```

Parameters

Inputs	Description	Base Type
Vi	Unique identifier for an IVI session	ViSession
<Name>Auto	If set to True, <Name> is automatically determined. If set to False, <Name> shall revert to the manual setting mode and use the value of the <Name> parameter. See the <Name> Auto attribute description for more details. The default value is "Off"	ViInt32
<Name>	[Adapt the description to the actual method.] If <Name> is set to a valid value, the <Name>Auto parameter is ignored and the <Name> Auto attribute is set to False.	ViInt32

Return Values (C/COM)

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

.NET Exceptions

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

6. Absolute Time (IVI-C and IVI-COM)

6.1 Overview

Instruments sometimes provide time stamps for measured data. In these cases it is necessary for drivers to include a means of setting and retrieving the absolute time. This section describes functions which shall be used in instrument classes for this purpose.

Note that for IVI.NET, absolute time is expressed using the PrecisionDateTime class.

6.1.1 Relationship to LXI-based instruments

IVI 3.15: IviLxiSync Specification includes techniques that allow instrument operation to be triggered at given times and for timestamps to be associated with measured data. For this, LXI instruments use the IEEE 1588 standard for high-precision time synchronization across all instruments in a test system. *IVI 3.15: IviLxiSync Specification* also specifies a particular data format (a pair of ViReal64 values) that is used to contain a high-resolution time stamp value. For compatibility, this data format is utilized here as well. However, the use of a compatible data format does not imply that the underlying time base is required to be compatible with the IEEE 1588 standard. Instruments are allowed to maintain their internal time base in any manner they wish.

Note that for .NET, these details are managed internally by the PrecisionDateTime and PrecisionTimeSpan classes.

The behavior of the Set Time function described here **is not necessarily the same as that defined for LXI devices.** To set the system time on instruments that implement IEEE 1588 synchronization, the time must be set on the system master clock. This specification does not address IEEE 1588 specifically, and the *IVI 3.15: IviLxiSync Specification* does not include a method that allows users to set the system time. At the discretion of the vendor, some implementations of the Set Time function may always affect the IEEE 1588 system time; some implementations may affect the system time if the driver is controlling the instrument that contains IEEE 1588 master clock; and other implementations may not affect the system time at all.

Conversely, the Get Time function defined here is identical to the Get System Time function in the *IVI 3.15: IviLxiSync Specification*. On LXI devices, the behavior of these two functions shall be identical.

6.2 Attributes

Time-based capabilities use the following attributes:

- Time (IVI.NET Only)

This section describes the behavior and requirements of this attribute.

6.2.1 Time (IVI.NET Only)

Data Type	Access	Applies to	Coercion	High Level Functions
`Ivi.Driver.PrecisionDateTime`	R/W[4]	N/A	N/A	N/A

.NET Property Name

`Time`

COM Property Name

`N/A`

Use the Set Time and Get Time methods.

C Constant Name

`N/A`

Use the Set Time and Get Time methods.

Description

The current time on the instrument. Units are implicit in the definition of PrecisionDateTime.

.NET Exceptions

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

[4] Access may be RO for classes or drivers where setting the property does not make sense. For example, RO access may be sufficient for instruments that determine time internal to the instrument.

6.3 Functions

Time-based capabilities use the following functions:

- Set Time (IVI-C and IVI-COM Only)
- Get Time (IVI-C and IVI-COM Only)

This section describes the behavior and requirements of these functions.

6.3.1 Set Time (IVI-C and IVI-COM Only)

Description

This function is used to set the current time on the instrument. Time is expressed in seconds since January 1, 1970 (the IEEE 1588 epoch 0). The time is determined by adding TimeSeconds and TimeFractional together.

Usage Note: The purpose of the TimeFractional parameter is to provide more resolution than TimeSeconds can provide. If more resolution is not needed, TimeFractional should be 0.0. If both parameters are needed, TimeSeconds should hold the integer portion of the time and TimeFractional the fractional portion.

.NET Method Prototype

N/A

Use the Time attribute.

COM Method Prototype

```
HRESULT SetTime([in] double TimeSeconds,
                [in] double TimeFractional);
```

C Prototype

```
ViStatus SetTime (ViSession Vi,
                  ViReal64 TimeSeconds,
                  ViReal64 TimeFractional);
```

Parameters

Inputs	Description	Data Type
Vi	Unique identifier for an IVI session.	ViSession

Outputs	Description	Data Type
TimeSeconds	The current time in seconds. If TimeSeconds is less than 0, the driver shall return E_IVI_INVALID_VALUE.	ViReal64
TimeFractional	Additional resolution for the current time in seconds. If TimeFractional is less than 0, the driver shall return E_IVI_INVALID_VALUE.	ViReal64

See *IVI-3.15: IviLxiSync Specification* for a more complete description of the two 64-bit real parameters.

Return Values

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

6.3.2 Get Time (IVI-C and IVI-COM Only)

Description

This function is used to retrieve the current time from the instrument. Time is expressed in seconds since January 1, 1970 (the IEEE 1588 epoch 0). The time is determined by adding TimeSeconds and TimeFractional together.

The purpose of the TimeFractional parameter is to provide more resolution than TimeSeconds can provide. If more resolution is not needed, TimeFractional may be 0.0. If both parameters are needed, TimeSeconds shall hold the integer portion of the time and TimeFractional the fractional portion.

.NET Method Prototype

```
N/A
```

Use the Time attribute.

COM Method Prototype

```
HRESULT Time.GetTime([in, out] double* TimeSeconds,
                     [in, out] double* TimeFractional);
```

C Prototype

```
ViStatus GetTime (ViSession Vi,
                  ViReal64* TimeSeconds,
                  ViReal64* TimeFractional);
```

Parameters

Inputs	Description	Data Type
Vi	Unique identifier for an IVI session.	ViSession

Outputs	Description	Data Type
TimeSeconds	The current time in seconds.	ViReal64
TimeFractional	Additional resolution for the current time in seconds.	ViReal64

Return Values

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