



IVI-3.3: Standard Cross-Class Capabilities Specification

November 17, 2008 Edition
Revision 2.0

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.

*Table
of
Contents*

1.	Overview of Standard Cross-Class Capabilities	5
1.1	Notation	5
1.2	When to Define a Standard Cross-Class Capability	5
1.3	When to Refer to a Standard Cross-Class Capability	5
2.	Software Triggering Capability	6
3.	Standard Trigger Source Values.....	7
4.	Repeated Capability Group	9
4.1	Overview.....	9
4.2	Attributes	10
4.2.1	<Capability> Item	10
4.2.2	<Capability> Count.....	11
4.2.3	Active <Capability>	11
4.2.4	<Capability> Name (COM only)	12
4.3	Functions.....	13
4.3.1	Get <Capability> Name (C only).....	13
4.3.2	Set Active <Capability>.....	14
5.	Absolute Time	14
5.1	Overview.....	14
5.1.1	Relationship to LXI-based instruments.....	14
5.2	Functions.....	15
5.2.1	Set Time	15
5.2.2	Get Time.....	16

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 0.1	September 15, 1999	Original draft.
Revision 0.2	February 8, 2000	
Revision 0.3	February 24, 2000	Changes based on comments at IVI meeting.
Revision 0.4	May 30, 2000	Changes based on comments at May 2000 IVI meeting. We removed the MultiPoint and IsOverRange sections.
Revision 0.5	November, 2000	Added a chapter to cover repeated capabilities.
Revision 0.6	July, 2001	Draft for final review process.
Revision 1.0vc2	September, 2001	Draft for final review process. 2 nd voting candidate
Revision 1.0vc3	December, 2001	Changes base on issues raised at the December IVI meeting.
Revision 1.0	April, 2002	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.

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.

COM Method Prototype

```
HRESULT SendSoftwareTrigger ();
```

C Function Prototype

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

Parameters

Inputs	Description
Vi	Instrument handle

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional standard cross-class status code:

- Trigger Not Software

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

<i>Error Name</i>	<i>Description</i>		
	<i>Language</i>	<i>Identifier</i>	<i>Value(hex)</i>
Trigger Not Software	The trigger source is not set to software trigger.		
	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.

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 three 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 both the IVI-C and IVI-COM APIs, the value may take the form of a string.

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_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_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"

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 Specification*
- *IVI 4-6: IviSwtch Class Specification*
- *IVI 4-7: IviPwrMeter Class Specification*
- *IVI 4-8: IviSpecAn Class Specification*
- *IVI 4-10: IviRFSigGen Class Specification*

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 3-2: API Style Guide*, describes three techniques for handling repeated capabilities. Each attribute or function includes a notation indicating whether it is used with a particular technique. Table 4-1 lists the techniques with the attributes and functions needed to implement that technique.

Table 4-1 Attributes and Functions used with various Repeated Capability Techniques

Technique	Attributes	Functions
Parameter	<Capability> Count <Capability> Name (COM only)	Get <Capability> Name (C only)
Selector	<Capability> Count Active <Capability> <Capability> Name (COM only)	Get <Capability> Name (C only) Set Active <Capability>
IVI-COM Collection	<Capability> Item <Capability> Count <Capability> Name	

These attributes and functions should be placed in the same capability group as the repeated capability with which they are associated. In the COM hierarchy, they are placed in the collection interface with the plural name. See Section 13.3, *COM Interface Hierarchy*, in *IVI 3-2: API Style Guide*.

This section uses the notation <capability> 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 <Capability> Count would appear as Channel Count in the instrument class. The plural form is shown as <Capability>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.

4.2 Attributes

Repeated capabilities use the following attributes:

- <Capability> Item
- <Capability> Count
- Active <Capability>
- Capability <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.

4.2.1 <Capability> Item

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

COM Property Name

```
<Capability>s.Item(BSTR Name);
```

COM Enumeration Name

N/A

C Constant Name

N/A

Description

Gets an interface pointer for an interface to control a particular <capability>. The name of the interface is controlled by the instrument class writers, but it should contain the name of class followed by the name of the capability.

Return Values

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

4.2.2 <Capability> Count

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

COM Property Name

<Capability>.s.Count

COM Enumeration Name

N/A

C Constant Name

PREFIX_ATTR_<CAPABILITY>_COUNT

Description

Specifies how many <capability>s are available.

Compliance Note

If the name of the repeated capability is Channel, then the attribute ID value for this attribute, PREFIX_ATTR_CHANNEL_COUNT, shall be IVI_INHERENT_ATTR_BASE + 203.

4.2.3 Active <Capability>

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

COM Property Name

Active<Capability>

COM Enumeration Name

N/A

C Constant Name*PREFIX_ATTR_ACTIVE_<CAPABILITY>***Description**

Specifies which <capability> is currently active. If the driver defines a qualified <capability> name, this attribute returns the qualified name.

Compliance Note**4.2.4 <Capability> Name (COM only)**

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

COM Property Name

Name ([in] LONG Index)

COM Enumeration Name

N/A

C Constant Name

N/A

Description

Returns the physical repeated capability identifier defined by the specific driver for the <capability> that corresponds to the one-based index that the user specifies. If the driver defines a qualified <capability> name, this attribute returns the qualified name. Valid values for the Index parameter are between one and the value of the <capability> Count attribute. If the user passes an invalid value for the Index parameter, the value of this attribute is an empty string.

4.3 Functions

Repeated capabilities use the following functions:

- Get <Capability> Name (C only)
- Set Active <Capability>

This section describes the behavior and requirements of these function.

4.3.1 Get <Capability> Name (C only)

Description

This function returns the physical name identifier defined by the specific driver for the <capability> that corresponds to the one-based index that the user specifies. If the driver defines a qualified <capability> 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 <capability> `Count`, the function returns an empty string in the `Name` parameter and returns an error.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_Get<capability>Name (ViSession Vi,  
                                              ViInt32 Index,  
                                              ViInt32 NameBufferSize,  
                                              ViChar Name[]);
```

Parameters

Inputs	Description	Base Type
<code>Vi</code>	Unique identifier for an IVI session	<code>ViSession</code>
<code>Index</code>	A one-based index that defines which name to return.	<code>ViInt32</code>
<code>NameBufferSize</code>	The number of bytes in the <code>ViChar</code> array that the user specifies for the <capability>Name parameter.	<code>ViInt32</code>

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

Return Values

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 <capability>, that is the <capability> Count attribute is one, the driver may return an empty string.
2. Refer to Section 3.1.2, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters*, in IVI-3.2 *Inherent Capabilities Specification*, Section 3.1.2.1 for rules regarding the NameBufferSize and Name parameters.

4.3.2 Set Active <Capability>

Description

This function sets the active <capability>.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_SetActive<Capability> (ViSession Vi,  
                                              ViString Name);
```

Parameters

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

Return Values

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

5. Absolute Time

5.1 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.

5.1.1 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.

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.

5.2 Functions

Time-based capabilities use the following functions:

- Set Time
- Get Time

This section describes the behavior and requirements of these functions.

5.2.1 Set Time

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.

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.

5.2.2 Get Time

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.

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.