



IVI-3.3: Standard Cross-Class Capabilities Specification

April, 2002
Revision 1.0

Important Information

This specification (IVI-3.3) is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at www.ivifoundation.org, or contact the IVI Foundation at 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.

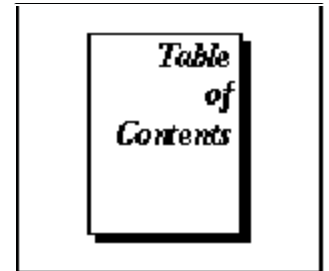
The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation via email at ivilistserver@ivifoundation.org, via the web site at www.ivifoundation.org, or you can write to the IVI Foundation, 2515 Camino del Rio South, Suite 340, San Diego, CA 92108.

Warranty

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

Trademarks

Product and company names listed are trademarks or trade names of their respective companies. No investigation has been made of common-law trademark rights in any work.



1.	Overview of 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.	Repeated Capability Group	7
3.1	Overview	7
3.2	Attributes	8
3.2.1	<Capability> Item.....	8
3.2.2	<Capability> Count.....	8
3.2.3	Active <Capability>.....	9
3.2.4	<Capability> Name (COM only).....	9
3.3	Functions	11
3.3.1	Get <Capability> Name (C only).....	11
3.3.2	Set Active <Capability>.....	12

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

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 specifications 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. Repeated Capability Group

3.1 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 3-1 lists the techniques with the attributes and functions needed to implement that technique.

Table 3-1 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.

3.2 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.

3.2.1 <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.

3.2.2 <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.

3.2.3 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.

Compliance Note**3.2.4 <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. 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.

3.3 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.

3.3.1 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 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.

3.3.2 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.

Compliance Notes

3. For an instrument with only one <capability>, that is the <capability> Count attribute is one, the driver may return an empty string.
4. See *IVI-3.2 Inherent Capabilities Specification*, Section 3.1.2.1 for additional rules for C Functions with ViChar Array Output Parameters.