



IVI-3.2: Inherent Capabilities Specification

March 23, 2009 Edition
Revision 1.3

Important Information

The IVI-3.2: Inherent 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.

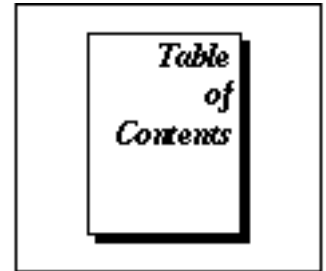
Warranty

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.



| | |
|--|-----------|
| Inherent Capabilities Specification..... | 6 |
| 1. Overview of the Inherent Capabilities Specification..... | 8 |
| 1.1 Introduction..... | 8 |
| 1.2 Inherent Capabilities Overview | 8 |
| 1.3 References..... | 8 |
| 1.4 Definitions of Terms and Acronyms..... | 8 |
| 2. Specification Section Layout | 9 |
| 2.1 Introduction..... | 9 |
| 2.1.1 Attribute Section Layout..... | 9 |
| 2.1.2 Function Section Layout..... | 9 |
| 3. General Requirements | 10 |
| 3.1 Inherent Capabilities Compliance Rules..... | 10 |
| 3.1.1 Attribute Compliance Rules..... | 10 |
| 3.1.2 Function Compliance Rules..... | 10 |
| 3.1.2.1 Additional Compliance Rules for C Functions with ViChar Array Output Parameters | 11 |
| 3.1.2.2 Additional Compliance Rules for Revision String Attributes | 11 |
| 3.1.3 Boolean Attribute and Parameter Values..... | 12 |
| 4. Inherent Capabilities Overview | 13 |
| 4.1 COM Inherent Capabilities | 13 |
| 4.1.1 Inherent Capabilities Interfaces | 14 |
| 4.1.1.1 IIVIvDriver..... | 14 |
| 4.1.1.2 IIVIvDriverOperation..... | 14 |
| 4.1.1.3 IIVIvComponentIdentity | 14 |
| 4.1.1.4 IIVIvDriverIdentity | 15 |
| 4.1.1.5 IIVIvDriverUtility | 15 |
| 4.1.2 Interface Reference Properties..... | 15 |
| 4.1.2.1 Driver Identity | 15 |
| 4.1.2.2 Driver Operation..... | 15 |
| 4.1.2.3 Driver Utility | 15 |
| 4.1.3 IIVIvDriver COM Category..... | 16 |
| 4.2 C Inherent Capabilities | 17 |
| 4.3 Relationship of Inherent Attributes and Different Types of IIVIvDriver..... | 20 |

5. Inherent Property/Attribute Descriptions21

| | |
|--|----|
| 5.1 Cache | 21 |
| 5.2 Class Driver Class Spec Major Version (IVI-C Only)..... | 22 |
| 5.3 Class Driver Class Spec Minor Version (IVI-C Only)..... | 23 |
| 5.4 Class Driver Description (IVI-C Only)..... | 24 |
| 5.5 Class Driver Prefix (IVI-C Only)..... | 25 |
| 5.6 Class Driver Revision (IVI-C Only) | 26 |
| 5.7 Class Driver Vendor (IVI-C Only) | 27 |
| 5.8 Class Group Capabilities..... | 28 |
| 5.9 Component Class Spec Major Version (IVI-COM Only)..... | 29 |
| 5.10 Component Class Spec Minor Version (IVI-COM Only)..... | 30 |
| 5.11 Component Description (IVI-COM Only)..... | 31 |
| 5.12 Component Identifier (IVI-COM Only)..... | 32 |
| 5.13 Component Revision (IVI-COM Only) | 33 |
| 5.14 Component Vendor (IVI-COM Only) | 34 |
| 5.15 Driver Setup..... | 35 |
| 5.16 I/O Resource Descriptor..... | 36 |
| 5.17 Initialized (IVI-COM Only)..... | 37 |
| 5.18 Instrument Firmware Revision..... | 38 |
| 5.19 Instrument Manufacturer..... | 39 |
| 5.20 Instrument Model..... | 40 |
| 5.21 Interchange Check | 41 |
| 5.22 Logical Name..... | 43 |
| 5.23 Query Instrument Status..... | 44 |
| 5.24 Range Check | 45 |
| 5.25 Record Value Coercions | 46 |
| 5.26 Simulate | 47 |
| 5.27 Specific Driver Class Spec Major Version (IVI-C Only) | 48 |
| 5.28 Specific Driver Class Spec Minor Version (IVI-C Only) | 49 |
| 5.29 Specific Driver Description (IVI-C Only) | 50 |
| 5.30 Specific Driver Locator (IVI-C Only)..... | 51 |
| 5.31 Specific Driver Prefix (IVI-C Only) | 52 |
| 5.32 Specific Driver Revision (IVI-C Only)..... | 53 |
| 5.33 Specific Driver Vendor (IVI-C Only)..... | 54 |
| 5.34 Supported Instrument Models..... | 55 |

6. Inherent Method/Function Descriptions56

| | |
|---|----|
| 6.1 Clear Error (IVI-C Only) | 56 |
| 6.2 Clear Interchange Warnings..... | 57 |
| 6.3 Close | 58 |
| 6.4 Disable | 59 |
| 6.5 Error Message (IVI-C Only)..... | 60 |
| 6.6 Error Query | 61 |
| 6.7 Get Attribute <type> (IVI-C Only)..... | 63 |
| 6.8 Get Attribute ViString (IVI-C Only)..... | 65 |
| 6.9 Get Error (IVI-C Only) | 66 |
| 6.10 Get Next Coercion Record..... | 68 |
| 6.11 Get Next Interchange Warning | 70 |
| 6.12 Get Specific Driver C Handle (IVI-C Only) | 72 |
| 6.13 Get Specific Driver IUnknown Pointer (IVI-C Only)..... | 73 |
| 6.14 Initialize | 74 |
| 6.15 Invalidate All Attributes..... | 79 |
| 6.16 Lock Session | 80 |
| 6.17 Reset | 82 |

| | |
|--|------------|
| 6.18 Reset Interchange Check..... | 83 |
| 6.19 Reset With Defaults | 84 |
| 6.20 Revision Query (IVI-C Only) | 85 |
| 6.21 Self Test..... | 87 |
| 6.22 Set Attribute <type> (IVI-C Only)..... | 89 |
| 6.23 Unlock Session | 91 |
| 7. Specific Driver Wrapper Functions..... | 92 |
| 7.1 C Wrappers for IVI-COM Specific Drivers | 93 |
| 7.1.1 Get Native IUnknown Pointer (IVI-C Only) | 93 |
| 7.1.2 Attach To Existing COM Session (IVI-C Only)..... | 94 |
| 7.2 COM Wrappers for IVI-C Specific Drivers | 95 |
| 7.2.1 Native C Handle (IVI-COM Only)..... | 95 |
| 7.2.2 Attach To Existing C Session (IVI-COM Only)..... | 96 |
| 8. IVI Inherent Attribute ID Definitions | 97 |
| 8.1 Inherent Attribute ID Values..... | 97 |
| 8.2 Reserved Vendor Specific Inherent Extension Attribute ID Values and Constants..... | 98 |
| 8.3 Reserved Module Private Attribute ID Values | 99 |
| 8.4 Reserved Standard Cross Class Capabilities Attribute ID Values | 99 |
| 9. Common Error and Completion Codes | 100 |
| 9.1 IVI Error and Completion Codes | 100 |
| 9.2 IVI-C Error and Completion Codes | 102 |
| 9.3 IVI-COM Error and Completion Codes..... | 105 |
| 9.4 Reserved Vendor Specific Error and Completion Code Values and Constants | 107 |
| 9.5 Standard COM Error Codes for Use during Driver Development | 109 |
| 9.6 Unused Standard COM Error Codes..... | 109 |
| Appendix A. ANSI C Include File..... | 110 |
| Appendix B. COM IDL File | 114 |
| B.1 IviDriverTypeLib.idl | 114 |
| B.2 IviDriver.idl..... | 115 |
| B.3 IviDriverEnglish.idl..... | 121 |

Inherent Capabilities Specification

Revision History

This section is an overview of the revision history of the Inherent Capabilities specification. Specific individual additions/modifications to the document in draft revisions are denoted with diff-marks, “[”], in the right hand column of each line of text to which the change/modification applies.

Table 1. Inherent Capabilities Specification Revisions

| Revision Number | Date of Revision | Revision Notes |
|------------------|-------------------|--|
| Revision 0.3 | January 11, 1998 | Original draft. |
| Revision 0.4 | April 1, 1999 | Updated to include all required functions and attributes. |
| Revision 0.5 | January, 2000 | Changed to include changes to attribute and function names made as part of IVI COM working group activity |
| Revision 0.6 | November, 2000 | Changed to include changes to attribute and function names made as part of IVI COM working group activity. Added standard error and completion codes for C and COM. |
| Revision 0.7 | January, 2001 | Reformatted standard error and completion codes for C and COM. Miscellaneous content and format changes from November 2000 meeting. |
| Revision 0.8 | May, 2001 | Changed to reflect decisions made at the February 2001 meeting. |
| Revision 1.0vc1a | November 19, 2001 | Continuous updates and edits for final voting |
| Revision 1.0vc1b | February 7, 2002 | Removed locator property; removed circular references; minor updates from reviews. IVI 3.1 related changes include: revision, new error codes, references to IVI 3.5, and clarified Error Query content. |
| Revision 1.0vc1c | March 5, 2002 | Updated TOC. Fixed a document map problem. Checked section references in other specs. Updated IviDriverEnglish.IDL with typo corrections to help strings. |
| Revision 1.0 | April 15, 2002 | Accepted all changes. Removed change tracking. Removed draft comment and changed version to 1.0. |
| Revision 1.0 | October 1, 2004 | IVI-COM drivers do not support multithread locks on sessions. |
| Revision 1.1 | January 12, 2007 | Added attribute accessor functions for 64-bit integers. |
| Revision 1.2 | November 17, 2008 | Added a requirement that 64-bit drivers include a specific string (identifying the driver as 64-bit) in the values for the following property/attributes: |

Table 1. Inherent Capabilities Specification Revisions

| | | |
|--------------|----------------|--|
| | | <p>IviComponentIdentity “Description” property for IVI-COM, CLASS_DRIVER_DESCRIPTION attribute for IVI-C, SPECIFIC_DRIVER_DESCRIPTION attribute for IVI-C.</p> <p>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.</p> |
| Revision 1.3 | March 23, 2009 | Added a note in Sections 6.7 and 6.22 that an IVI-C specific driver may omit the ViInt64 function if the driver has no 64-bit attributes. |

1. Overview of the Inherent Capabilities Specification

1.1 Introduction

This section summarizes *Inherent Capabilities Specification* itself and contains general information that might help the reader understand, interpret, and implement aspects of this specification. The contents of this section include the following:

- Inherent Capabilities Overview
- The definitions of terms and acronyms
- References

1.2 Inherent Capabilities Overview

This specification defines the capabilities that all IVI instrument drivers are required to implement. This includes IVI-COM and IVI-C drivers, as well as COM and C wrappers for native IVI-C and IVI-COM drivers.

For a complete description of the various types of IVI drivers, refer to *IVI-3.1: Driver Architecture Specification*.

1.3 References

Several other documents and specifications are related to this specification. These other related documents are as follows:

- IVI-1: Charter Document
- IVI-3.1: Driver Architecture Specification
- IVI-3.5: Configuration Server Specification
- VPP-3.3: Instrument Driver Interactive Developer Interface Specification
- VPP-4.3.2: VISA Implementation Specification for Textual Language
- VPP-4.3.4: VISA Implementation Specification for COM

1.4 Definitions of Terms and Acronyms

Refer to *IVI-5: Glossary* for a description of the terms and acronyms used in this specification. This specification does not define any additional terms.

2. Specification Section Layout

2.1 Introduction

This section gives an overview of the information presented for each property/attribute and method/function that this specification defines.

2.1.1 Attribute Section Layout

Each Attribute section is composed of the following subsections. Optional subsections are noted:

Capabilities Table—A table that defines the following for the attribute:

Data Type—Specifies the *VXIplug&play* data type of the attribute. Possible values include `ViInt32`, `ViInt64`, `ViReal64`, `ViBoolean`, `ViString`, and `ViSession`.

Access—Specifies the kind of access the user has to the attribute. Possible values are Read-Only, Write-Only, and Read/Write.

RO (Read-Only)—indicates that the user can only get the value of the attribute.

WO (Write-Only)—indicates that the user can only set the value of the attribute.

R/W (Read/Write)—indicates that the user can get and set the value of the attribute.

COM Property Name—Defines the property name, including the object hierarchy, that an IVI-COM specific driver uses for the property.

C Constant Name—Defines the attribute name that an IVI-C driver uses for the attribute. To determine the actual C constant name for a particular IVI-C driver, replace the literal string `PREFIX` with the macro prefix for the IVI-C driver.

Description—Describes the attribute and its intended use.

Defined Values (Optional)—Defines all the valid values for the attribute.

Compliance Notes (Optional)—Section 3, *General Requirements*, defines the general rules an IVI driver shall follow to be compliant with an attribute specification. This section specifies additional compliance requirements and exceptions that apply to a particular attribute.

2.1.2 Function Section Layout

Each Function section is composed of the following subsections. Optional subsections are noted:

Description—Describes the behavior and intended use of the function.

COM Method Prototype—Defines the COM method prototype, including the object hierarchy, that an IVI-COM specific driver uses for the method.

C Function Prototype—Defines the prototype that an IVI-C driver uses for the function. To determine the actual C function name for a particular IVI-C driver, replace the literal string `Prefix` with the function prefix for the IVI-C driver.

Parameters—Describes each function parameter.

Return Values—Defines the possible completion codes for the function.

Compliance Notes (Optional)—Section 3, *General Requirements*, defines the general rules an IVI driver shall follow to be compliant with a function specification. This section specifies additional compliance requirements and exceptions that apply to a particular function.

3. General Requirements

This section describes the general requirements an IVI driver shall meet to be compliant with this specification.

3.1 Inherent Capabilities Compliance Rules

To comply with this specification, an IVI driver shall comply with the following rules:

- Implement all attributes that this specification defines, except when noted otherwise.
- Implement all functions that this specification defines, except when noted otherwise.

3.1.1 Attribute Compliance Rules

To comply with a particular attribute that this specification defines, an IVI driver shall adhere to the compliance rules defined in Section 5.6.1, *Attribute Compliance Rules*, of the *IVI-3.1: Driver Architecture Specification*.

In addition, the IVI driver shall adhere to all of the following requirements for the attribute:

- Implement the attribute as non-channel based.
- Implement the attribute with no value coercions. Value coercions are not allowed for inherent attributes. IVI drivers shall report an error if the IVI driver or the instrument cannot accept the value that the user specifies for an inherent attribute.

Note: If a particular attribute has compliance rules or exceptions in addition to the rules that this section defines, the *Compliance Notes* section for the attribute contains the additional rules or exceptions.

3.1.2 Function Compliance Rules

To comply with a particular function that this specification defines, an IVI driver shall adhere to the compliance rules defined in Section 5.6.2, *Function Compliance Rules*, of the *IVI-3.1: Driver Architecture Specification*.

Note: If a particular function has compliance rules or exceptions in addition to the rules that this section defines, the *Compliance Notes* section for the function contains the additional rules or exceptions.

3.1.2.1 Additional Compliance Rules for C Functions with ViChar Array Output Parameters

This section specifies additional compliance rules for C functions that have a `ViChar` array output parameter and an input parameter that specifies the size of the `ViChar` array. The functions in this specification that have such parameters are the Get Attribute ViString, Get Error, Get Next Coercion Record, and Get Next Interchange Warning functions.

- The user is responsible for allocating a `ViChar` array and passing the address of the array in the `ViChar` array output parameter. The array serves as a buffer into which the IVI-C driver copies a string.
- The name of the input parameter that specifies the size of the array is the name of the array followed by `BufferSize` and is the parameter that immediately precedes the `ViChar` array output parameter. For example if the name of the `ViChar` array output parameter is `errorDescription`, the name of the buffer size parameter is `errorDescriptionBufferSize`. The user passes the number of bytes in the buffer as the buffer size parameter.
- If the string that the function attempts to copy contains more bytes (including the terminating NUL byte) than the user indicates in the buffer size parameter, the function does the following:
 - Copies (buffer size-1) bytes into the buffer
 - Places an ASCII NUL byte at the end of the buffer
 - Returns in the return value the buffer size that the user must pass to get the entire string.

For example, if the value is `123456` and the buffer size is 4, the function places `123` followed by a NUL byte into the buffer and returns 7. If the function encounters an error, the function returns the corresponding error code instead of the required buffer size.

- If the user passes a negative number for the buffer size parameter, the function copies the value into the buffer regardless of the number of bytes in the value.
- If the user passes 0 for the `BufferSize` parameter, the function allows the user to pass `VI_NULL` for the output buffer parameter and returns the buffer size that the user must pass to get the entire value including the NUL byte.

Note: The preceding compliance rules regarding `ViChar` array output parameters and corresponding buffer size parameters do not apply to the Self Test, Revision Query, Error Query, and Error Message functions. These functions do not have buffer size parameters.

3.1.2.2 Additional Compliance Rules for Revision String Attributes

This section specifies additional compliance rules for attributes that return revision strings. The attributes in this specification that return revision strings are Component Revision, Class Driver Revision, and Specific Driver Revision.

The revision string that these attributes return is in the following format:

```
revision[ string]
```

The format of the `revision` shall follow the rules for `FileVersion` defined in Section 5.18, *File Versioning*, in *IVI-3.1: Driver Architecture Specification*. The `string` is optional. If the `string` is present, a space shall separate the `revision` from the `string`. The `string` contains additional driver specific revision information. Multi-byte characters are not allowed in the string that this attribute returns. String characters shall be in the range of (`\x20-\x7E`).

Examples of allowed revision strings are shown below:

4.00

02.0001.12345.1 This revision adds XYZ capability to the component

3.1.3 Boolean Attribute and Parameter Values

This specification uses True and False as the values for Boolean attributes and parameters. The following table defines the identifiers that are used for True and False in the IVI-COM and IVI-C architectures.

| Boolean Value | IVI-COM Identifier | IVI-C Identifier |
|----------------------|---------------------------|-------------------------|
| True | VARIANT_TRUE | VI_TRUE |
| False | VARIANT_FALSE | VI_FALSE |

4. Inherent Capabilities Overview

This section gives an overview of the inherent capabilities of IVI-COM and IVI-C drivers. The inherent capabilities for an IVI-COM driver consist of a set of properties and methods. The inherent capabilities for an IVI-C driver consist of a set of attributes and functions. In most cases, the COM properties and methods have corresponding C attributes and functions. This section defines a *generic name* for each property/attribute combination and method/function combination. The remainder of this specification uses the generic name to refer to properties/attributes and methods/functions.

4.1 COM Inherent Capabilities

The following table shows the inherent capabilities of an IVI-COM driver. The COM Interface Hierarchy specifies the relationship of the inherent properties and methods for IVI-COM drivers. The Generic Name column lists the generic name for each property or method. The Type column uses a P or an M to specify whether the item is a property or method.

Table 4-1. Inherent Capabilities of an IVI-COM Driver

| COM Interface Hierarchy | Generic Name | Type |
|----------------------------|------------------------------------|------|
| Close | Close | M |
| DriverOperation | | |
| Cache | Cache | P |
| ClearInterchangeWarnings | Clear Interchange Warnings | M |
| DriverSetup | Driver Setup | P |
| GetNextCoercionRecord | Get Next Coercion Record | M |
| GetNextInterchangeWarning | Get Next Interchange Warning | M |
| InterchangeCheck | Interchange Check | P |
| InvalidateAllAttributes | Invalidate All Attributes | M |
| LogicalName | Logical Name | P |
| QueryInstrumentStatus | Query Instrument Status | P |
| RangeCheck | Range Check | P |
| RecordCoercions | Record Value Coercions | P |
| ResetInterchangeCheck | Reset Interchange Check | M |
| IoResourceDescriptor | I/O Resource Descriptor | P |
| Simulate | Simulate | P |
| Identity | | |
| Description | Component Description | P |
| Identifier | Component Identifier | P |
| Revision | Component Revision | P |
| Vendor | Component Vendor | P |
| InstrumentManufacturer | Instrument Manufacturer | P |
| InstrumentModel | Instrument Model | P |
| InstrumentFirmwareRevision | Instrument Firmware Revision | P |
| SpecificationMajorVersion | Component Class Spec Major Version | P |
| SpecificationMinorVersion | Component Class Spec Minor Version | P |

Table 4-1. Inherent Capabilities of an IVI-COM Driver

| COM Interface Hierarchy | Generic Name | Type |
|---------------------------|-----------------------------|------|
| SupportedInstrumentModels | Supported Instrument Models | P |
| GroupCapabilities | Class Group Capabilities | P |
| Initialize | Initialize | M |
| Initialized | Initialized | P |
| Utility | | |
| Disable | Disable | M |
| ErrorQuery | Error Query | M |
| LockObject | Lock Session | M |
| Reset | Reset | M |
| ResetWithDefaults | Reset With Defaults | M |
| SelfTest | Self Test | M |
| UnlockObject | Unlock Session | M |

4.1.1 Inherent Capabilities Interfaces

IVI-COM inherent capabilities are organized into six COM interfaces. Table 4-2 lists the interfaces and their GUIDs.

Table 4-2. Inherent Capabilities COM Interface GUIDs

| Interface | GUID |
|------------------------|--|
| IIVI-Driver | {47ed5184-a398-11d4-ba58-000064657374} |
| IIVI-DriverOperation | {47ed5188-a398-11d4-ba58-000064657374} |
| IIVI-ComponentIdentity | {47ed5185-a398-11d4-ba58-000064657374} |
| IIVI-DriverIdentity | {47ed5186-a398-11d4-ba58-000064657374} |
| IIVI-DriverUtility | {47ed5189-a398-11d4-ba58-000064657374} |

4.1.1.1 IIVI-Driver

IIVI-Driver is the root interface for all IVI-COM drivers. It contains methods and properties that initialize, close, and query the state of the IVI driver session. It also contains three interface reference properties. Refer to Section 4.1.2, *Interface Reference Properties*, for more information.

4.1.1.2 IIVI-DriverOperation

IIVI-DriverOperation contains methods and properties that manage the operation of the driver.

4.1.1.3 IIVI-ComponentIdentity

IIVI-ComponentIdentity contains properties that return general information related to the identity of an IVI component.

4.1.1.4 IIVIriverIdentity

IIVIriverIdentity inherits from IIVIComponentIdentity. It adds properties that return information related to the identity of the driver and of the instrument.

4.1.1.5 IIVIriverUtility

IIVIriverUtility contains methods that provide a basic set of utility operations.

4.1.2 Interface Reference Properties

Interface reference properties are used to navigate the COM Inherent Capabilities hierarchy. Refer to Section 5.15.3, *IVI-COM Inherent Interfaces* in *IVI-3.1: Driver Architecture Specification*, for more information on interface reference properties. This section describes the interface reference properties that the IIVIriver interface defines.

4.1.2.1 Driver Identity

| Data Type | Access |
|--------------------|--------|
| IIVIriverIdentity* | RO |

COM Property Name

Identity

Description

Returns a pointer to the IIVIriverIdentity interface.

4.1.2.2 Driver Operation

| Data Type | Access |
|---------------------|--------|
| IIVIriverOperation* | RO |

COM Property Name

DriverOperation

Description

Returns a pointer to the IIVIriverOperation interface.

4.1.2.3 Driver Utility

| Data Type | Access |
|-------------------|--------|
| IIVIriverUtility* | RO |

COM Property Name

Utility

Description

Returns a pointer to the IIVIUtility interface.

4.1.3 IIVI COM Category

The COM Category for inherent capabilities shall be “IIVI”, and the Category ID (CATID) shall be {47ed5152-a398-11d4-ba58-000064657374 }.

4.2 C Inherent Capabilities

Unlike COM inherent capabilities, the C inherent capabilities consist of separate hierarchies of attributes and functions. The hierarchy of C inherent attributes is shown in the following table.

The Category or Generic Attribute Name column shows how the various inherent attributes are divided into categories and specifies the generic name for each attribute. The C Defined Constant column gives the C constant name for each attribute. The COM Interface column lists the COM interface location of the corresponding COM property. N/A in the COM Interface column specifies that the attribute does not have a corresponding COM property.

For IVI-C drivers, the *prefix.sub* file must implement the attribute hierarchy as shown in this table.

Table 4-2. Hierarchy of C Inherent Attributes

| Category or Generic Attribute Name | C Defined Constant | COM Interface |
|--|---|-----------------|
| <i>Inherent IVI Attributes</i> | | |
| <i>User Options</i> | | |
| Range Check | <i>PREFIX_ATTR_RANGE_CHECK</i> | DriverOperation |
| Query Instrument Status | <i>PREFIX_ATTR_QUERY_INSTRUMENT_STATUS</i> | DriverOperation |
| Cache | <i>PREFIX_ATTR_CACHE</i> | DriverOperation |
| Simulate | <i>PREFIX_ATTR_SIMULATE</i> | DriverOperation |
| Record Value Coercions | <i>PREFIX_ATTR_RECORD_COERCIONS</i> | DriverOperation |
| Interchange Check | <i>PREFIX_ATTR_INTERCHANGE_CHECK</i> | DriverOperation |
| <i>Class Driver Identification</i> | | |
| Class Driver Description | <i>PREFIX_ATTR_CLASS_DRIVER_DESCRIPTION</i> | N/A |
| Class Driver Prefix | <i>PREFIX_ATTR_CLASS_DRIVER_PREFIX</i> | N/A |
| Class Driver Vendor | <i>PREFIX_ATTR_CLASS_DRIVER_VENDOR</i> | N/A |
| Class Driver Revision | <i>PREFIX_ATTR_CLASS_DRIVER_REVISION</i> | N/A |
| Class Driver Class Spec Major Version | <i>PREFIX_ATTR_CLASS_DRIVER_CLASS_SPEC_MAJOR_VERSION</i> | N/A |
| Class Driver Class Spec Minor Version | <i>PREFIX_ATTR_CLASS_DRIVER_CLASS_SPEC_MINOR_VERSION</i> | N/A |
| <i>Driver Identification</i> | | |
| Specific Driver Description | <i>PREFIX_ATTR_SPECIFIC_DRIVER_DESCRIPTION</i> | N/A |
| Specific Driver Prefix | <i>PREFIX_ATTR_SPECIFIC_DRIVER_PREFIX</i> | N/A |
| Specific Driver Locator | <i>PREFIX_ATTR_SPECIFIC_DRIVER_LOCATOR</i> | N/A |
| Specific Driver Vendor | <i>PREFIX_ATTR_SPECIFIC_DRIVER_VENDOR</i> | N/A |
| Specific Driver Revision | <i>PREFIX_ATTR_SPECIFIC_DRIVER_REVISION</i> | N/A |
| Specific Driver Class Spec Major Version | <i>PREFIX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION</i> | N/A |
| Specific Driver Class Spec Minor Version | <i>PREFIX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION</i> | N/A |
| <i>Driver Capabilities</i> | | |

Table 4-2. Hierarchy of C Inherent Attributes

| Category or Generic Attribute Name | C Defined Constant | COM Interface |
|-------------------------------------|---|-----------------|
| Supported Instrument Models | <i>PREFIX_ATTR_SUPPORTED_INSTRUMENT_MODELS</i> | Identity |
| Class Group Capabilities | <i>PREFIX_ATTR_GROUP_CAPABILITIES</i> | Identity |
| <i>Instrument Identification</i> | | |
| Instrument Manufacturer | <i>PREFIX_ATTR_INSTRUMENT_MANUFACTURER</i> | Identity |
| Instrument Model | <i>PREFIX_ATTR_INSTRUMENT_MODEL</i> | Identity |
| Instrument Firmware Revision | <i>PREFIX_ATTR_INSTRUMENT_FIRMWARE_REVISION</i> | Identity |
| <i>Advanced Session Information</i> | | |
| Logical Name | <i>PREFIX_ATTR_LOGICAL_NAME</i> | DriverOperation |
| I/O Resource Descriptor | <i>PREFIX_ATTR_IO_RESOURCE_DESCRIPTOR</i> | DriverOperation |
| Driver Setup | <i>PREFIX_ATTR_DRIVER_SETUP</i> | DriverOperation |

Note: IVI-C specific drivers do not implement or export the Class Driver Description, Class Driver Prefix, Class Driver Vendor, Class Driver Revision, Class Driver Class Spec Major Version, and Class Driver Class Spec Minor Version attributes.

The hierarchy of C inherent functions is shown in the following table. The Category or Generic Function Name column lists the generic name for each function and divides the functions into categories. The C Function Name lists the C function names. The COM Interface column lists the COM interface location of the corresponding COM method. N/A in the COM Interface column specifies that the function does not have a corresponding COM method.

Note: If an IVI driver contains a Configure category in its function hierarchy, then the Attribute Access Function category must be a sub-category of the Configure category.

Table 4-3. Hierarchy of C Inherent Functions

| Category or Generic Function Name | C Function Name | COM Interface |
|--------------------------------------|--|-----------------|
| Initialize | <i>Prefix_init</i> | N/A |
| Initialize With Options | <i>Prefix_InitWithOptions</i> | Main |
| <i>Attribute Access Functions</i> | | |
| Set Attribute Functions | <i>Prefix_SetAttribute<type></i> | N/A |
| Get Attribute Functions | <i>Prefix_GetAttribute<type></i> | N/A |
| Invalidate All Attributes | <i>Prefix_InvalidateAllAttributes</i> | DriverOperation |
| <i>Utility Functions</i> | | |
| Self Test | <i>Prefix_self_test</i> | Utility |
| Reset | <i>Prefix_reset</i> | Utility |
| ResetWithDefaults | <i>Prefix_ResetWithDefaults</i> | Utility |
| Disable | <i>Prefix_Disable</i> | Utility |
| Revision Query | <i>Prefix_revision_query</i> | N/A |
| Error Query | <i>Prefix_error_query</i> | Utility |
| Error Message | <i>Prefix_error_message</i> | N/A |
| Get Specific Driver C Handle | <i>Prefix_GetSpecificDriverCHandle</i> | N/A |
| Get Specific Driver IUnknown Pointer | <i>Prefix_GetSpecificDriverIUnknownPtr</i> | N/A |
| Get Error | <i>Prefix_GetError</i> | N/A |
| Clear Error | <i>Prefix_ClearError</i> | N/A |
| Lock Session | <i>Prefix_LockSession</i> | Utility |
| Unlock Session | <i>Prefix_UnlockSession</i> | Utility |
| Get Next Coercion Record | <i>Prefix_GetNextCoercionRecord</i> | DriverOperation |
| Get Next Interchange Warning | <i>Prefix_GetNextInterchangeWarning</i> | DriverOperation |
| Reset Interchange Check | <i>Prefix_ResetInterchangeCheck</i> | DriverOperation |
| Clear Interchange Warnings | <i>Prefix_ClearInterchangeWarnings</i> | DriverOperation |
| Close | <i>Prefix_close</i> | Main |

Note: Initialize With Options is a variation of the Initialize function and is discussed in Section 6.14, *Initialize*.

Note: IVI-C specific drivers do not implement or export the Get Specific Driver C Handle and Get Specific Driver IUnknown Pointer functions.

4.3 Relationship of Inherent Attributes and Different Types of IVI Driver

Some inherent attributes are exported by all types of IVI drivers—IVI-COM specific drivers, IVI-C specific drivers, and IVI class drivers. Other inherent attributes are exported by only one or two types of drivers. Generally, inherent attributes fall into the following two categories:

- Attributes that are exported by IVI-COM specific drivers, IVI-C specific drivers, and IVI class drivers. When the user accesses this type of attribute through an IVI class driver, the IVI class driver sets or returns the value of the attribute in the IVI specific driver. Examples are Cache, Supported Instrument Models, Instrument Manufacturer, and Logical Name.
- Attributes whose Generic names start with Component, Specific Driver, or Class Driver. These attributes generally come in threes, for example, Component Description, Specific Driver Description, and Class Driver Description. For this categories of attributes, the following general rules apply:
 - IVI-COM specific drivers export only the attributes whose names start with Component.
 - IVI-C specific drivers export only the attributes whose names start with Specific Driver.
 - IVI-C class drivers export both the attributes whose names start with Specific Driver and those that start with Class Driver. The attributes whose names start with Class Driver return information about the IVI-C class driver. The attributes whose names start with Specific Driver return information about the IVI specific driver. Thus, the user of the IVI-C class driver can get information about both the IVI-C class driver and the IVI specific driver.

In general, IVI-C specific drivers and IVI-C class drivers export the attributes whose names start with Specific Driver. An exception is the Specific Driver Locator attribute. Only IVI-C class drivers export this attribute, not IVI-C specific drivers because they cannot reliably determine their own location.

5. Inherent Property/Attribute Descriptions

This section gives a complete description of each inherent property/attribute.

5.1 Cache

| Data Type | Access |
|-----------|--------|
| ViBoolean | R/W |

COM Property Name

`DriverOperation.Cache`

C Constant Name

`PREFIX_ATTR_CACHE`

Description

Specifies whether or not to cache the value of attributes. When caching is enabled, the IVI specific driver keeps track of the current instrument settings so that it can avoid sending redundant commands to the instrument.

The default value is True. When the user opens an instrument session through an IVI class driver or uses a logical name to initialize a specific driver, the user can override this value by specifying a value in the IVI configuration store. The Initialize function allows the user to override both the default value and the value that the user specifies in the IVI configuration store.

Defined Values

| Value | Description |
|-------|---|
| True | The specific driver caches the value of attributes. |
| False | The specific driver does not cache the value of attributes. |

Compliance Notes

1. The IVI specific driver shall accept both the True and False values for this attribute.
2. For each attribute, the IVI specific driver developer can choose whether caching is always enabled, caching is always disabled, or whether caching is configurable by the user. If the specific driver has attributes for which caching is configurable by the user, the specific driver caches the values of these attributes when the Cache attribute is set to True and does not cache values when the Cache attribute is set to False.

5.2 Class Driver Class Spec Major Version (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViInt32 | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_CLASS_DRIVER_CLASS_SPEC_MAJOR_VERSION`

Description

Returns the major version number of the IVI class specification in accordance with which the IVI-C class driver was developed. The value is a positive integer value.

Compliance Notes

1. IVI specific drivers shall not implement or export this attribute.
2. IVI-C class drivers shall set the value of this attribute.

5.3 Class Driver Class Spec Minor Version (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViInt32 | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_CLASS_DRIVER_CLASS_SPEC_MINOR_VERSION`

Description

Returns the minor version number of the IVI class specification in accordance with which the IVI-C class driver was developed. The value is a non-negative integer value.

Compliance Notes

1. IVI specific drivers shall not implement or export this attribute.
2. IVI-C class drivers shall set the value of this attribute.

5.4 Class Driver Description (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_CLASS_DRIVER_DESCRIPTION`

Description

Returns a brief description of the IVI-C class driver.

If the driver is compiled for use in 64-bit applications, the description shall include the following statement at the end identifying it as 64-bit.

[Compiled for 64-bit.]

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

Compliance Notes

1. IVI specific drivers shall not implement or export this attribute.
2. IVI-C class drivers shall set the value of this attribute.

5.5 Class Driver Prefix (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_CLASS_DRIVER_PREFIX`

Description

Returns the case sensitive prefix of the user-callable functions that the IVI-C class driver exports.

The name of each user-callable function in the class driver begins with this prefix. For example, if a class driver has a user-callable function named `IviDmm_init`, then `IviDmm` is the prefix for that driver.

The string that this attribute returns contains a maximum of 32 bytes including the NUL byte.

Compliance Notes

1. IVI specific drivers shall not implement or export this attribute.
2. IVI-C class drivers shall set the value of this attribute.

5.6 Class Driver Revision (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_CLASS_DRIVER_REVISION`

Description

Returns version information about the IVI-C class driver. Refer to Section 3.1.2.2, *Additional Compliance Rules for Revision String Attributes*, for additional rules regarding this attribute.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

Compliance Notes

1. IVI specific drivers shall not implement or export this attribute.
2. IVI-C class drivers shall set the value of this attribute.

5.7 Class Driver Vendor (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

PREFIX_ATTR_CLASS_DRIVER_VENDOR

Description

Returns the name of the vendor that supplies the IVI-C class driver.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

Compliance Notes

1. IVI specific drivers shall not implement or export this attribute.
2. IVI-C class drivers shall set the value of this attribute.

5.8 Class Group Capabilities

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

Identity.GroupCapabilities

C Constant Name

PREFIX_ATTR_GROUP_CAPABILITIES

Description

Returns a comma-separated list that identifies the class capability groups that the IVI specific driver implements. The items in the list are capability group names that the IVI class specifications define. The string has no white space except for white space that might be embedded in a capability group name.

If the IVI specific driver does not comply with an IVI class specification, the specific driver returns an empty string as the value of this attribute.

The string that this attribute returns does not have a predefined maximum length.

5.9 Component Class Spec Major Version (IVI-COM Only)

| Data Type | Access |
|-----------|--------|
| ViInt32 | RO |

COM Property Name

`Identity.SpecificationMajorVersion`

C Constant Name

N/A

Description

Returns the major version number of the class specification in accordance with which the IVI-COM software component was developed. The value is a positive integer value.

If the software component is not compliant with a class specification, the software component returns zero as the value of this attribute.

5.10 Component Class Spec Minor Version (IVI-COM Only)

| Data Type | Access |
|-----------|--------|
| ViInt32 | RO |

COM Property Name

`Identity.SpecificationMinorVersion`

C Constant Name

N/A

Description

Returns the minor version number of the class specification in accordance with which the IVI-COM software component was developed. The value is a non-negative integer value.

If the software component is not compliant with a class specification, the software component returns zero as the value of this attribute.

5.11 Component Description (IVI-COM Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

Identity.Description

C Constant Name

N/A

Description

Returns a brief description of the IVI-COM software component.

If the driver is compiled for use in 64-bit applications, the description shall include the following statement at the end identifying it as 64-bit.

[Compiled for 64-bit.]

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.12 Component Identifier (IVI-COM Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

Identity.Identifier

C Constant Name

N/A

Description

Returns the case-sensitive unique identifier of the IVI-COM software component.

The string that this attribute returns contains a maximum of 32 bytes including the NUL byte.

5.13 Component Revision (IVI-COM Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

Identity.Revision

C Constant Name

N/A

Description

Returns version information about the IVI-COM software component. Refer to Section 3.1.2.2, *Additional Compliance Rules for Revision String Attributes*, for additional rules regarding this attribute.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.14 Component Vendor (IVI-COM Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

Identity.Vendor

C Constant Name

N/A

Description

Returns the name of the vendor that supplies the IVI-COM software component.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.15 Driver Setup

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

`DriverOperation.DriverSetup`

C Constant Name

`PREFIX_ATTR_DRIVER_SETUP`

Description

Returns the driver setup string that the user specified in the IVI configuration store when the instrument driver session was initialized or passes in the `OptionString` parameter of the `Initialize` function. Refer to Section 6.14, *Initialize*, for the restrictions on the format of the driver setup string.

The string that this attribute returns does not have a predefined maximum length.

5.16 I/O Resource Descriptor

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

`DriverOperation.IoResourceDescriptor`

C Constant Name

`PREFIX_ATTR_IO_RESOURCE_DESCRIPTOR`

Description

Returns the resource descriptor that the user specified for the physical device. The user specifies the resource descriptor by editing the IVI configuration store or by passing a resource descriptor to the Initialize function of the specific driver. Refer to Section 6.14, *Initialize*, for the restrictions on the contents of the resource descriptor string.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

Compliance Notes

1. If the resource descriptor *is not* available while simulating, the IVI specific driver returns an empty string.
2. If the resource descriptor *is* available while simulating, the IVI specific driver returns it.

5.17 Initialized (IVI-COM Only)

| Data Type | Access |
|-----------|--------|
| ViBoolean | RO |

COM Property Name

Initialized

C Constant Name

N/A

Description

Returns a value that indicates whether the IVI-COM specific driver is in the initialized state. After the specific driver is instantiated and before the Initialize function successfully executes, this attribute returns False. After the Initialize function successfully executes and prior to the execution of the Close function, this attribute returns True. After the Close function executes, this attribute returns False.

The Initialized attribute is one of the few IVI-COM specific driver attributes that can be accessed while the specific driver is not in the initialized state. All the attributes of an IVI-COM specific driver that can be accessed while the specific driver is not in the initialized state are listed below.

- Component Class Spec Major Version
- Component Class Spec Minor Version
- Component Description
- Component Prefix
- Component Identifier
- Component Revision
- Component Vendor
- Initialized
- Supported Instrument Models

5.18 Instrument Firmware Revision

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

`Identity.InstrumentFirmwareRevision`

C Constant Name

`PREFIX_ATTR_INSTRUMENT_FIRMWARE_REVISION`

Description

Returns an instrument specific string that contains the firmware revision information of the physical instrument. The IVI specific driver returns the value it queries from the instrument as the value of this attribute.

In some cases, it is not possible for the specific driver to query the firmware revision of the instrument. This can occur when the Simulate attribute is set to True or if the instrument is not capable of returning the firmware revision. For these cases, the specific driver returns defined strings for this attribute. If the Simulate attribute is set to True, the specific driver returns `Not available while simulating` as the value of this attribute. If the instrument is not capable of returning the firmware version and the Simulate attribute is set to False, the specific driver returns `Cannot query from instrument` as the value of this attribute.

The string that this attribute returns does not have a predefined maximum length.

5.19 Instrument Manufacturer

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

`Identity.InstrumentManufacturer`

C Constant Name

`PREFIX_ATTR_INSTRUMENT_MANUFACTURER`

Description

Returns the name of the manufacturer of the instrument. The IVI specific driver returns the value it queries from the instrument as the value of this attribute.

In some cases, it is not possible for the IVI specific driver to query the manufacturer of the instrument. This can occur when the Simulate attribute is set to True or if the instrument is not capable of returning the manufacturer's name. For these cases, the specific driver returns defined strings for this attribute. If the Simulate attribute is set to True, the specific driver returns `Not available while simulating` as the value of this attribute. If the instrument is not capable of returning the manufacturer name and the Simulate attribute is set to False, the specific driver returns `Cannot query from instrument` as the value of this attribute.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.20 Instrument Model

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

`Identity.InstrumentModel`

C Constant Name

`PREFIX_ATTR_INSTRUMENT_MODEL`

Description

Returns the model number or name of the physical instrument. The IVI specific driver returns the value it queries from the instrument as the value of this attribute.

In some cases, it is not possible for the IVI specific driver to query the model number of the instrument. This can occur when the Simulate attribute is set to True or if the instrument is not capable of returning the model number. For these cases, the specific driver returns defined strings for this attribute. If the Simulate attribute is set to True, the specific driver returns `Not available while simulating` as the value of this attribute. If the instrument is not capable of returning the model number and the Simulate attribute is set to False, the specific driver returns `Cannot query from instrument` as the value of this attribute.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.21 Interchange Check

| Data Type | Access |
|-----------|--------|
| ViBoolean | R/W |

COM Property Name

`DriverOperation.InterchangeCheck`

C Constant Name

`PREFIX_ATTR_INTERCHANGE_CHECK`

Description

Specifies whether the IVI specific driver performs interchangeability checking. If the Interchange Check attribute is enabled, the specific driver maintains a record of each interchangeability warning that it encounters. The user calls the Get Next Interchange Warning function to extract and delete the oldest interchangeability warning from the list. Refer to Section 6.11, *Get Next Interchange Warning*, Section 6.2, *Clear Interchange Warnings*, and Section 6.18, *Reset Interchange Check*, for more information.

If the user opens an instrument session through an IVI class driver and the Interchange Check attribute is enabled, the IVI class driver may perform additional interchangeability checking. The IVI class driver maintains a list of the interchangeability warnings that it encounters. The user can retrieve both class driver interchangeability warnings and specific driver interchangeability warnings by calling the Get Next Interchange Warning function on the class driver session.

If the IVI specific driver does not implement interchangeability checking, the specific driver returns the Value Not Supported error when the user attempts to set the Interchange Check attribute to True. If the specific driver does implement interchangeability checking and the user opens an instrument session through an IVI class driver, the IVI class driver accepts True as a valid value for the Interchange Check attribute even if the class driver does not implement interchangeability checking capabilities of its own.

The default value is False. If the user opens an instrument session through an IVI class driver or initializes an IVI specific driver with a logical name, the user can override this value in the IVI configuration store. The Initialize function allows the user to override both the default value and the value that the user specifies in the IVI configuration store.

Defined Values

| Value | Description |
|-------|---|
| True | The specific driver performs interchangeability checking. |
| False | The specific driver does not perform interchangeability checking. |

Compliance Notes

1. An IVI specific driver shall accept False as a valid value for this attribute.
2. If an IVI specific driver implements True as a valid value for this attribute, then the specific driver shall implement the interchangeability checking rules that the corresponding class specification defines.
3. If an IVI specific driver implements True as a valid value for this attribute, then the specific driver shall implement the Get Next Interchange Warning, Reset Interchange Check, and Clear Interchange Warnings functions.

4. An IVI driver can impose a restriction on the number of interchangeability warnings that the driver records in the list. If the driver imposes a restriction, the driver shall throw away the oldest interchangeability warning in the list when the driver attempts to record a new interchangeability warning and the list is full.

5.22 Logical Name

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

`DriverOperation.LogicalName`

C Constant Name

`PREFIX_ATTR_LOGICAL_NAME`

Description

Returns the IVI logical name that the user passed to the Initialize function. If the user initialized the IVI specific driver directly and did not pass a logical name, then this attribute returns an empty string. Refer to *IVI-3.5: Configuration Server Specification* for restrictions on the format of IVI logical names.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.23 Query Instrument Status

| Data Type | Access |
|-----------|--------|
| ViBoolean | R/W |

COM Property Name

`DriverOperation.QueryInstrumentStatus`

C Constant Name

`PREFIX_ATTR_QUERY_INSTRUMENT_STATUS`

Description

Specifies whether the IVI specific driver queries the instrument status at the end of each user operation. Querying the instrument status is very useful for debugging. After validating the program, the user can set this attribute to False to disable status checking and maximize performance. The user specifies this value for the entire IVI driver session.

The default value is False. When the user opens an instrument session through an IVI class driver or uses a logical name to initialize an IVI specific driver, the user can override this value by specifying a value in the IVI configuration store. The Initialize function allows the user to override both the default value and the value that the user specifies in the IVI configuration store.

Defined Values

| Value | Description |
|-------|--|
| True | The IVI specific driver queries the instrument status after each operation. |
| False | The IVI specific driver does not query the instrument status after each operation. |

Compliance Notes

1. The IVI specific driver shall implement both the True and False values for this attribute.
2. If the instrument status can be queried for its status and this attribute is set to True, then the IVI specific driver checks the instrument status at the end of every call by the user to a function that accesses the instrument.
3. If the instrument status cannot be queried independently of user operations, then this attribute has no effect on the behavior of the IVI specific driver.

5.24 Range Check

| Data Type | Access |
|-----------|--------|
| ViBoolean | R/W |

COM Property Name

`DriverOperation.RangeCheck`

C Constant Name

`PREFIX_ATTR_RANGE_CHECK`

Description

Specifies whether the IVI specific driver validates attribute values and function parameters. If enabled, the specific driver validates the parameter values that users pass to driver functions. Validating attribute values and function parameters is useful for debugging. After validating the program, the user can set this attribute to False to disable range checking and maximize performance.

The default value is True. When the user opens an instrument session through an IVI class driver or uses a logical name to initialize an IVI specific driver, the user can override this value by specifying a value in the IVI configuration store. The Initialize function allows the user to override both the default value and the value that the user specifies in the IVI configuration store.

Defined Values

| Value | Description |
|-------|---|
| True | The IVI specific driver validates attribute values and function parameters. |
| False | The IVI specific driver does not validate attribute values and function parameters. |

Compliance Notes

1. The IVI specific driver shall implement both the True and False values for this attribute.
2. Regardless of the value to which the user sets this attribute, the IVI specific driver is not required to duplicate all range checking operations that the instrument firmware performs.
3. If this attribute is set to False, the IVI specific driver does not perform range-checking operations that the specific driver developer considers non-essential and time consuming.

5.25 Record Value Coercions

| Data Type | Access |
|-----------|--------|
| ViBoolean | R/W |

COM Property Name

`DriverOperation.RecordCoercions`

C Constant Name

`PREFIX_ATTR_RECORD_COERCIONS`

Description

Specifies whether the IVI specific driver keeps a list of the value coercions it makes for `ViInt32` and `ViReal64` attributes. If the Record Value Coercions attribute is enabled, the specific driver maintains a record of each coercion. The user calls the Get Next Coercion Record function to extract and delete the oldest coercion record from the list. Refer to Section 6.10, *Get Next Coercion Record*, for more information.

If the IVI specific driver does not implement coercion recording, the specific driver returns the Value Not Supported error when the user attempts to set the Record Value Coercions attribute to True.

The default value is False. When the user opens an instrument session through an IVI class driver or uses a logical name to initialize a IVI specific driver, the user can override this value by specifying a value in the IVI configuration store. The Initialize function allows the user to override both the default value and the value that the user specifies in the IVI configuration store.

Defined Values

| Value | Description |
|-------|---|
| True | The IVI specific driver records coercion information. |
| False | The IVI specific driver does not record coercion information. |

Compliance Notes

1. The IVI specific driver shall accept False as a valid value for this attribute.
2. If an IVI specific driver implements True as a valid value for this attribute, then the specific driver shall implement the Get Next Coercion Record function.
3. The IVI specific driver can impose a restriction on the number of coercion records that the specific driver records in the list. If the specific driver imposes a restriction, the specific driver shall throw away the oldest coercion record in the list when the specific driver attempts to record a new coercion record and the list is full.

5.26 Simulate

| Data Type | Access |
|-----------|--------|
| ViBoolean | R/W |

COM Property Name

`DriverOperation.Simulate`

C Constant Name

`PREFIX_ATTR_SIMULATE`

Description

Specifies whether or not the IVI specific driver simulates instrument driver I/O operations. If simulation is enabled, the specific driver functions do not perform instrument I/O. For output parameters that represent instrument data, the specific driver functions return simulated values.

The default value is False. When the user opens an instrument session through an IVI class driver or uses a logical name to initialize an IVI specific driver, the user can override this value by specifying a value in the IVI configuration store. The Initialize function allows the user to override both the default value and the value that the user specifies in the IVI configuration store.

Defined Values

| Value | Description |
|-------|--|
| True | The IVI specific driver simulates instrument driver I/O. |
| False | The IVI specific driver does not simulate instrument driver I/O. |

Compliance Notes

1. The IVI specific driver shall implement both the True and False values for this attribute.
2. When Simulate is set to True, the IVI specific driver may perform less rigorous range checking operations than when Simulate is set to False.
3. If the IVI specific driver is initialized with Simulate set to True, the specific driver shall return the Cannot Change Simulation State error if the user attempts to set Simulate to False prior to calling the Close function.

5.27 Specific Driver Class Spec Major Version (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViInt32 | RO |

COM Property Name

N/A

C Constant Name

PREFIX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION

Description

Returns the major version number of the class specification in accordance with which the IVI specific driver was developed. The value is a positive integer value.

If the IVI specific driver is not compliant with a class specification, the specific driver returns zero as the value of this attribute.

5.28 Specific Driver Class Spec Minor Version (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViInt32 | RO |

COM Property Name

N/A

C Constant Name

PREFIX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION

Description

Returns the minor version number of the class specification in accordance with which the IVI specific driver was developed. The value is a non-negative integer value.

If the IVI specific driver is not compliant with a class specification, the specific driver returns zero as the value of this attribute.

5.29 Specific Driver Description (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_SPECIFIC_DRIVER_DESCRIPTION`

Description

Returns a brief description of the IVI specific driver.

If the driver is compiled for use in 64-bit applications, the description shall include the following statement at the end identifying it as 64-bit.

[Compiled for 64-bit.]

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.30 Specific Driver Locator (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_SPECIFIC_DRIVER_LOCATOR`

Description

Returns the location of the IVI specific driver software module. The user identifies the specific driver by passing a logical name to the Initialize function of the class driver. The user configures the location of the specific driver in the IVI configuration store.

If the class driver instantiates an underlying IVI-COM class-compliant specific driver, the value of this property is the COM class ID (CLSID) of the underlying IVI-COM specific driver object that implements the root class-compliant interface. The string returned always has exactly 36 characters, with a format of 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX', where X is a valid hexadecimal digit.

If the class driver instantiates an underlying IVI-C class-compliant specific driver, the value of this property is the full DLL pathname of underlying IVI-C specific driver that implements the root class-compliant interface. The string returned in this case may be of arbitrary length.

If the underlying IVI-COM class-compliant specific driver does not implement a class-compliant interface that is recognized by the IVI-C class driver, the IVI-C class driver returns an empty string for this attribute.

Refer to *IVI-3.5: Configuration Server Specification* for more information regarding the possible values of this attribute.

Compliance Notes

1. IVI specific drivers shall not implement or export this attribute.
2. IVI-C class drivers shall set the value of this attribute.

5.31 Specific Driver Prefix (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_SPECIFIC_DRIVER_PREFIX`

Description

Returns the case-sensitive prefix of the user-callable functions that the IVI-C specific driver exports. For an IVI-C specific driver, the name of each user-callable function in the specific driver begins with this prefix. For example, if the Fluke 45 driver has a user-callable function named `f145_init`, then `f145` is the prefix for that driver.

The string that this attribute returns contains a maximum of 32 bytes including the NUL byte.

5.32 Specific Driver Revision (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

`PREFIX_ATTR_SPECIFIC_DRIVER_REVISION`

Description

Returns version information about the IVI specific driver. Refer to Section 3.1.2.2, *Additional Compliance Rules for Revision String Attributes*, for additional rules regarding this attribute.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.33 Specific Driver Vendor (IVI-C Only)

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

N/A

C Constant Name

PREFIX_ATTR_SPECIFIC_DRIVER_VENDOR

Description

Returns the name of the vendor that supplies the IVI specific driver.

The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

5.34 Supported Instrument Models

| Data Type | Access |
|-----------|--------|
| ViString | RO |

COM Property Name

`Identity.SupportedInstrumentModels`

C Constant Name

`PREFIX_ATTR_SUPPORTED_INSTRUMENT_MODELS`

Description

Returns a comma-separated list of names of instrument models with which the IVI specific driver is compatible. The string has no white space except possibly embedded in the instrument model names. An example of a string that this attribute might return is `TKTDS3012,TKTDS3014,TKTDS3016`.

It is not necessary for the string to include the abbreviation for the manufacturer if it is the same for all models. In the example above, it is valid for the attribute to return the string `TDS3012,TDS3014,TDS3016`.

The string that this attribute returns does not have a predefined maximum length.

6. Inherent Method/Function Descriptions

This section gives a complete description of each inherent method/function.

6.1 Clear Error (IVI-C Only)

Description

This function clears the error code and error description for the current execution thread and for the IVI session. If the user specifies a valid IVI session for the `vi` parameter, this function clears the error information for the session. If the user passes `VI_NULL` for the `vi` parameter, this function clears the error information for the current execution thread. If the `vi` parameter is an invalid session, the function does nothing and returns an error.

The function clears the error code by setting it to `IVI_SUCCESS`. If the error description string is non-NULL, the function de-allocates the error description string and sets the address to `VI_NULL`.

Maintaining the error information separately for each thread is useful if the user does not have a session handle to pass to the `Prefix_GetError`, `Prefix_ClearError`, or `Prefix_error_message` function, which occurs when a call to `Initialize` fails.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_ClearError (ViSession Vi);
```

Parameters

| Inputs | Description | Data Type |
|-----------------|--|------------------------|
| <code>vi</code> | Unique identifier for an IVI session. The user can pass <code>VI_NULL</code> . | <code>ViSession</code> |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

6.2 Clear Interchange Warnings

Description

This function clears the list of interchangeability warnings that the IVI specific driver maintains.

When this function is called on an IVI class driver session, the function clears the list of interchangeability warnings that the class driver and the specific driver maintain.

Refer to the Interchange Check attribute for more information on interchangeability checking.

COM Method Prototype

```
HRESULT DriverOperation.ClearInterchangeWarnings();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_ClearInterchangeWarnings (ViSession Vi);
```

Parameters

| Inputs | Description | Datatype |
|--------|---------------------------------------|-----------|
| vi | Unique identifier for an IVI session. | ViSession |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. If an IVI-COM specific driver does not accept True as a valid value for the Interchange Check attribute, then the IVI-COM specific driver shall return the Function Not Supported error when the user calls this function.
2. If an IVI-C specific driver does not accept True as a valid value for the Interchange Check attribute, then the IVI-C specific driver shall not export this function.

6.3 Close

Description

When the user finishes using an IVI driver session, the user must call the Close function. This function closes the I/O session to the instrument. This function may put the instrument into an idle state before closing the I/O session.

For IVI-COM specific drivers, this function also does the following:

- Prevents the user from calling other functions in the driver that access the instrument until the user calls the Initialize function again.
- May deallocate internal resources used by the IVI session.

For IVI-C specific drivers, this function also does the following:

- Destroys the IVI session and all its attributes.
- Deallocates any memory resources used by the IVI session.

COM Method Prototype

```
HRESULT Close();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_close (ViSession Vi);
```

Parameters

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

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. It is possible for a user to perform the following sequence of operations on an IVI specific driver:
 - Call the Initialize function with Simulate set to False
 - Programmatically set Simulate to True
 - Call the Close function with Simulate still set to True

If this sequence occurs, the IVI specific driver shall execute the Close function as if Simulate was set to False.

6.4 Disable

Description

The Disable operation places the instrument in a quiescent state as quickly as possible. In a quiescent state, an instrument has no or minimal effect on the external system to which it is connected.

The Disable operation might be similar to the Reset operation in that it places the instrument in a known state. However, the Disable operation does not perform the other operations that the Reset operation performs such as configuring the instrument options on which the IVI specific driver depends. For some instruments, the disable function may do nothing.

The IVI class specifications define the exact behavior of this function for each instrument class. Refer to the IVI class specifications for more information on the behavior of this function.

COM Method Prototype

```
HRESULT Utility.Disable();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_Disable (ViSession Vi);
```

Parameters

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

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

6.5 Error Message (IVI-C Only)

Description

Translates the error return value from an IVI driver function to a user-readable string. This function returns the string that corresponds to the error code that the user passes in the `ErrorCode` parameter. The user can call this function at any time, without relation to a particular error occurrence.

The Error Message function shall accept a value of `VI_NULL` for the `Vi` input parameter. This allows the user to call the function even when Initialize fails.

When calling the Error Message function through a C interface, the user should pass a buffer with at least 256 bytes for the `ErrorMessage` parameter.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_error_message (ViSession Vi,  
                                       ViStatus ErrorCode,  
                                       ViChar ErrorMessage[]);
```

Parameters

| Inputs | Description | Data Type |
|------------------------|---------------------------------------|------------------------|
| <code>Vi</code> | Unique identifier for an IVI session. | <code>ViSession</code> |
| <code>ErrorCode</code> | Instrument driver status code | <code>ViStatus</code> |

| Outputs | Description | Data Type |
|---------------------------|---------------------------------|-----------------------|
| <code>ErrorMessage</code> | Instrument driver error message | <code>ViChar[]</code> |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. IVI-C specific drivers shall not write more than 256 characters, including the NUL byte, into the `ErrorMessage` output parameter.

6.6 Error Query

Description

Queries the instrument and returns instrument specific error information.

Generally, the user calls this function after another function in the IVI driver returns the Instrument Status error. The IVI specific driver returns the Instrument Status error when the instrument indicates that it encountered an error and its error queue is not empty. Error Query extracts an error out of the instrument's error queue.

For instruments that have status registers but no error queue, the IVI specific driver emulates an error queue in software.

When calling the Error Query function through a C interface, the user should pass a buffer with at least 256 bytes for the `ErrorMessage` parameter.

COM Method Prototype

```
HRESULT Utility.ErrorQuery([in,out] long* ErrorCode,  
                           [in,out] BSTR* ErrorMessage);
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_error_query (ViSession Vi,  
                                       ViInt32 * ErrorCode,  
                                       ViChar ErrorMessage[]);
```

Parameters

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

| Outputs | Description | Data Type |
|--------------|--------------------------|-----------|
| ErrorCode | Instrument error code | ViInt32 |
| ErrorMessage | Instrument error message | ViChar[] |

Return Values

The table below defines specific status codes that this function returns. Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

| Name | COM Identifier | C Identifier |
|---------------------------|---------------------------|-------------------------------|
| Error Query Not Supported | S_IVI_NSUP_ERROR_QUERY | IVI_WARN_NSUP_ERROR_QUERY |
| Unexpected Response | E_IVI_UNEXPECTED_RESPONSE | IVI_ERROR_UNEXPECTED_RESPONSE |

Compliance Notes

1. IVI-C specific drivers shall not write more than 256 characters, including the NUL byte, into the `ErrorMessage` output parameter.

2. The setting of the Query Instrument Status attribute shall have no effect on the operation of the Error Query function.

6.7 Get Attribute <type> (IVI-C Only)

Get Attribute ViInt32
Get Attribute ViInt64
Get Attribute ViReal64
Get Attribute ViBoolean
Get Attribute ViSession

Description

Obtains the current value of an attribute. A separate typesafe function exists for each possible attribute data type.

Notes:

1. A separate function description exists for Get Attribute ViString.
2. A specific driver may omit the ViInt64 function if the driver has no 64-bit attributes.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetAttributeViInt32 (ViSession vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViInt32 *AttributeValue);  
  
ViStatus _VI_FUNC Prefix_GetAttributeViInt64 (ViSession vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViInt64 *AttributeValue);  
  
ViStatus _VI_FUNC Prefix_GetAttributeViReal64 (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViReal64 *AttributeValue);  
  
ViStatus _VI_FUNC Prefix_GetAttributeViBoolean (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViBoolean *AttributeValue);  
  
ViStatus _VI_FUNC Prefix_GetAttributeViSession (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViSession *AttributeValue);
```

Parameters

| Inputs | Description | Data Type |
|------------------|--|---|
| Vi | Unique identifier for an IVI session. | ViSession |
| RepCapIdentifier | If the attribute is applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string. | ViConstString |
| AttributeID | The ID of the attribute. | ViAttr |
| AttributeValue | Returns the current value of the attribute. The user must specify the address of a variable that has the same data type as the attribute. | depends on the data type of the attribute |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

6.8 Get Attribute ViString (IVI-C Only)

Description

Obtains the current value of a ViString attribute.

Refer to Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters*, for additional rules regarding this function.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetAttributeViString (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViInt32 AttributeValueBufferSize,  
                                              ViChar AttributeValue[]);
```

Parameters

| Inputs | Description | Data Type |
|--------------------------|---|---------------|
| Vi | Unique identifier for an IVI session. | ViSession |
| RepCapIdentifier | If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string. | ViConstString |
| AttributeID | The ID of the attribute. | ViAttr |
| AttributeValueBufferSize | The number of bytes in the ViChar array that the user specifies for the AttributeValue parameter. | ViInt32 |

| Outputs | Description | Data Type |
|----------------|---|-----------|
| AttributeValue | The buffer in which the function returns the current value of the attribute. Can be VI_NULL if AttributeValueBufferSize is 0. | ViChar[] |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

6.9 Get Error (IVI-C Only)

Description

This function retrieves and then clears the IVI error information for the session or the current execution thread.

If the user specifies a valid IVI session for the `vi` parameter, Get Error retrieves and then clears the error information for the session. If the user passes `VI_NULL` for the `vi` parameter, Get Error retrieves and then clears the error information for the current execution thread. If the `vi` parameter is an invalid session, the function does nothing and returns an error. Normally, the error information describes the first error that occurred since the user last called the Get Error or Clear Error function.

One exception exists: If the `ErrorDescriptionBufferSize` parameter is zero, the function does not clear the error information. By passing 0 for the buffer size, the caller can ascertain the buffer size required to get the entire error description string and then call the function again with a sufficiently large buffer.

The precedence of errors and warnings is as follows:

- If there are no errors and no warnings, the IVI specific driver returns `IVI_SUCCESS` in the `ErrorCode` parameter and empty string in the `ErrorDescription` parameter.
- If there are warnings and no errors, the IVI specific driver returns the information regarding the first warning that it encountered.
- If there are errors, the IVI specific driver returns the information regarding the first error that it encountered.

The function complies with the rules in Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters*.

Note: IVI-COM specific drivers do not have a Get Error function because the information that the Get Error function returns is part of the COM error object.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetError (ViSession Vi,  
                                   ViStatus *ErrorCode,  
                                   ViInt32 ErrorDescriptionBufferSize,  
                                   ViChar ErrorDescription[]);
```

Parameters

| Inputs | Description | Data Type |
|--------------------------------|---|-----------|
| Vi | Unique identifier for an IVI session. The user can pass VI_NULL. | ViSession |
| ErrorDescription BufferSize | The number of bytes in the ViChar array that the user specifies for the ErrorDescription parameter. | ViInt32 |

| Outputs | Description | Data Type |
|------------------|--|-----------|
| ErrorCode | Returns the error code. Zero indicates that no error occurred. A positive value indicates a warning. A negative value indicates an error. The user can pass VI_NULL if the user does not want to retrieve this value. | ViStatus |
| ErrorDescription | Buffer into which the function copies the full formatted error string. The string describes the error code and any extra information regarding the error or warning condition. The buffer shall contain at least as many bytes as the user specifies in the ErrorDescriptionBufferSize parameter. The user can pass VI_NULL if the ErrorDescriptionBufferSize parameter is zero. | ViChar[] |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

6.10 Get Next Coercion Record

Description

If the Record Value Coercions attribute is set to True, the IVI specific driver keeps a list of all value coercions it makes on `ViInt32` or `ViReal64` attributes. This function obtains the coercion information associated with the IVI session. It retrieves and clears the oldest instance in which the specific driver coerced a value the user specified to another value.

The function returns an empty string in the `CoercionRecord` parameter if no coercion records remain for the session.

The following rules apply to the C interface of the Get Next Coercion Record function:

- The function complies with the rules in Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters*.
- If the user passes 0 for the `CoercionRecordBufferSize` parameter, the function does not clear a coercion record from the list.

The coercion record string shall contain the following information:

- The name of the attribute that was coerced. This can be the generic name, the COM property name, or the C defined constant.
- If the attribute applies to a repeated capability, the name of the virtual or physical repeated capability identifier.
- The value that the user specified for the attribute.
- The value to which the attribute was coerced.

A recommended format for the coercion record string is as follows:

```
"Attribute " + <attribute name> + [" on <repeated capability> " + <repeated capability identifier>] + " was coerced from " + <desiredVal> + " to " + <coercedVal>.
```

And example coercion record string is as follows:

```
Attribute TKTDS500_ATTR_VERTICAL_RANGE on channel ch1 was coerced from 9.0 to 10.0.
```

COM Method Prototype

```
HRESULT DriverOperation.GetNextCoercionRecord([out, retval] BSTR*  
CoercionRecord);
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetNextCoercionRecord (ViSession Vi,  
ViInt32 CoercionRecordBufferSize,  
ViChar CoercionRecord[]);
```

Parameters

| Inputs | Description | Data Type |
|--------------------------|---|-----------|
| Vi | Unique identifier for an IVI session. | ViSession |
| CoercionRecordBufferSize | The number of bytes in the ViChar array that the user specifies for the CoercionRecord parameter. | ViInt32 |

| Outputs | Description | Data Type |
|----------------|---|-----------|
| CoercionRecord | The buffer in which the function returns the oldest coercion record. Can be VI_NULL if CoercionRecordBufferSize is 0. | ViChar[] |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. If an IVI-COM specific driver does not accept True as a valid value for the Record Value Coercions attribute, then the IVI-COM specific driver shall return the Function Not Supported error when the user calls this function.
2. If an IVI-C specific driver does not accept True as a valid value for the Record Value Coercions attribute, then the IVI-C specific driver shall not export this function.

6.11 Get Next Interchange Warning

Description

If the Interchange Check attribute is set to True, the IVI specific driver keeps a list of all interchangeability warnings that it encounters. This function returns the interchangeability warnings associated with the IVI session. It retrieves and clears the oldest interchangeability warning from the list. Interchangeability warnings indicate that using the application with a different instrument might cause different behavior.

When this function is called on an IVI class driver session, it may return interchangeability warnings generated by the IVI class driver as well as interchangeability warnings generated by the IVI specific driver. The IVI class driver determines the relative order in which the IVI class driver warnings are returned in relation to the IVI specific driver warnings.

The function returns an empty string in the `InterchangeWarning` parameter if no interchangeability warnings remain for the session.

The following rules apply to the C interface of the Get Next Interchange Warning function:

- The function complies with the rules in Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters*.
- If the user passes 0 for the `InterchangeabilityWarningBufferSize` parameter, the function does not clear the oldest interchangeability warning from the list.

Refer to the Interchange Check attribute for more information on interchangeability checking.

COM Method Prototype

```
HRESULT DriverOperation.GetNextInterchangeWarning([out, retval] BSTR*
                                                    InterchangeWarning);
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetNextInterchangeWarning (ViSession Vi,
                                                    ViInt32 InterchangeWarningBufferSize,
                                                    ViChar InterchangeWarning[]);
```

Parameters

| Inputs | Description | Data Type |
|------------------------------|---|-----------|
| Vi | Unique identifier for an IVI session. | ViSession |
| InterchangeWarningBufferSize | The number of bytes in the ViChar array that the user specifies for the InterchangeWarning parameter. | ViInt32 |

| Outputs | Description | Data Type |
|--------------------|---|-----------|
| InterchangeWarning | The buffer in which the function returns the oldest interchange warning. Can be VI_NULL if InterchangeWarningBufferSize is 0. | ViChar[] |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. If an IVI-COM specific driver does not accept True as a valid value for the Interchange Check attribute, then the IVI-COM specific driver shall return the Function Not Supported error when the user calls this function.
2. If an IVI-C specific driver does not accept True as a valid value for the Interchange Check attribute, then the IVI-C specific driver shall not export this function.

6.12 Get Specific Driver C Handle (IVI-C Only)

Description

Returns a C session handle for the IVI specific driver that the IVI class driver is currently using. After the user retrieves the C session handle from the from the class driver, the user can pass the handle as the IVI session to the IVI-C specific driver. This enables the user to access the class driver for the portions of the program that are interchangeable and then access instrument specific functions or attributes in the IVI-C specific driver for the portions of the program that require instrument specific functionality.

If the class driver currently has a C session handle for the specific driver, the class driver returns that handle.

If the class driver does not have a C session handle for the specific driver and the specific driver has an IVI-C wrapper, the class driver attempts to open a C session through the C wrapper. If successful, the class driver returns the C session handle that it obtains from the wrapper.

If the IVI specific driver cannot be accessed through a C interface, the IVI class driver returns zero as the value of the handle.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetSpecificDriverCHandle (ViSession Vi,  
                                                  ViSession *SpecificDriverCHandle);
```

Parameters

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

| Outputs | Description | Data Type |
|-----------------------|---|-----------|
| SpecificDriverCHandle | Returns the C session handle of the IVI-C specific driver that the IVI class driver is currently using. | ViSession |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. IVI specific drivers shall not implement or export this function.

6.13 Get Specific Driver IUnknown Pointer (IVI-C Only)

Description

Returns an IUnknown pointer for the IVI specific driver that the IVI class driver is currently using. After the user retrieves the IUnknown pointer from the class driver, the user can use this value to access the IVI-COM specific driver. This enables the user to access the class driver for the portions of the program that are interchangeable and then access instrument specific functions or attributes in the IVI-COM specific driver for the portions of the program that require instrument specific functionality.

If the class driver currently has an IUnknown pointer for the specific driver, the class driver returns that pointer.

If the class driver does not have an IUnknown pointer for the specific driver and the specific driver has an IVI-COM wrapper, the class driver attempts to create a COM object through the IVI-COM wrapper. If successful, the class driver returns the IUnknown pointer that it obtains from the wrapper.

If the IVI specific driver cannot be accessed through a COM interface, the IVI class driver returns zero as the value of the IUnknown pointer.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetSpecificDriverIUnknownPtr (ViSession Vi,  
IUnknown **SpecificDriverIUnknownPtr);
```

Parameters

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

| Outputs | Description | Data Type |
|---------------------------|---|-----------|
| SpecificDriverIUnknownPtr | Returns the IUnknown pointer to the IVI specific driver that the class driver is currently using. | IUnknown |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. IVI specific drivers shall not implement or export this function.

6.14 Initialize

Description

The user must call the Initialize function prior to calling other IVI driver functions that access the instrument. When using an IVI-C specific driver, the user must call the Initialize function prior to calling *any* other instrument driver functions. A few exceptions exist. The user can call the Error Message, Get Error, and Clear Error functions and pass `VI_NULL` for the `vi` parameter prior to calling the Initialize function.

If simulation is disabled when the user calls the Initialize function, the function performs the following actions:

- Opens and configures an I/O session to the instrument.
- If the user passes `True` for the `IdQuery` parameter, the function queries the instrument for its ID and verifies that the IVI specific driver supports the particular instrument model. If the instrument cannot return its ID, the specific driver returns the ID Query Not Supported warning.
- If the user passes `True` for the `Reset` parameter, the function places the instrument in a known state. In an IEEE 488.2 instrument, the function sends the command string `*RST` to the instrument. If the instrument cannot perform a reset, the IVI specific driver returns the Reset Not Supported warning.
- Configures instrument options on which the IVI specific driver depends. For example, a specific driver might enable or disable headers or enable binary mode for waveform transfers.
- Performs the following operations in the given order:
 - Disables the class extension capability groups that the IVI specific driver implements.
 - If the class specification with which the IVI specific driver is compliant defines initial values for attributes, this function sets the attributes to the values that the class specification defines.
 - If the `ResourceName` parameter is a logical name, the IVI specific driver configures the initial settings for the specific driver and instrument based on the configuration of the logical name in the IVI configuration store.
- If the user accesses the IVI specific driver through its C interface, then the specific driver performs the following additional operations.
 - Creates a new IVI driver session.
 - Returns a `viSession` handle that identifies the session in subsequent calls to the IVI driver.

If simulation is enabled when the user calls the Initialize function, the function performs the following actions:

- If the user passes `True` for the `IdQuery` parameter and the instrument cannot return its ID, the IVI specific driver returns the ID Query Not Supported warning.
- If the user passes `True` for the `Reset` parameter and the instrument cannot perform a reset, the IVI specific driver returns the Reset Not Supported warning.
- If the `ResourceName` parameter is a logical name, the IVI specific driver configures the initial settings for the specific driver based on the configuration of the logical name in the IVI configuration store.
- If the user accesses the IVI specific driver through its C interface, then the specific driver performs the following additional operations.
 - Creates a new IVI driver session.
 - Returns a `viSession` handle that identifies the session in subsequent calls to the IVI driver.

Some instrument driver operations require or take into account information from the IVI configuration store. Examples of such information are virtual repeated capability name mappings and the value of certain inherent attributes. An IVI driver shall retrieve all the information for a session from the IVI configuration store during the Initialization function. The IVI driver shall not read any information from the IVI configuration store for a session after the Initialization function completes. Refer to Section 3.2.3, *Instantiating the Right Configuration Store From Software Modules*, of *IVI-3.5: Configuration Server Specification* for details on how to correctly instantiate the configuration store.

The `ResourceName` parameter must contain either a logical name that is defined in the IVI configuration store or an instrument specific string that identifies the I/O address of the instrument, such as a VISA resource descriptor string. Refer to *IVI-3.5: Configuration Server Specification* for restrictions on the format of IVI logical names. Refer to the *VXIplug&play* specifications for the grammar of VISA resource descriptor strings. Valid values for the `ResourceName` parameter depend on how the user initializes the session:

- After instantiating an IVI-COM specific driver through the IVI Factory, the user is expected to pass to the Initialize function the same logical name that the user passed to the IVI Factory. If the user passes a different logical name, the behavior of the driver is undefined.
- After instantiating an IVI-COM specific driver directly, the user can pass either a logical name or a resource descriptor to the Initialize function.
- When using the Initialize function to open a session to an IVI-C class driver, the user must pass a logical name.
- When using the Initialize function to open a session to an IVI-C specific driver, the user can pass either a logical name or a resource descriptor.

The user can use the `OptionsString` parameter to specify the initial values of certain IVI inherent attributes for the session. Table 6-1 lists the inherent attributes that the user can set through the `OptionsString` parameter. The user does not have to specify all or any of the attributes in the options string. If the user does not specify the initial value of an inherent attribute in the `OptionsString` parameter, the initial value of the attribute depends on the value of the `ResourceName` parameter:

- If the `ResourceName` parameter contains an IVI logical name, the IVI specific driver configures the initial settings based on the configuration of the logical name in the IVI configuration store.
- If the `ResourceName` parameter contains a resource descriptor string that identifies the I/O address of the instrument, the IVI specific driver sets inherent attributes to their *default initial values*. Table 6-1 shows the default initial value for each attribute.

The following table lists the IVI inherent attributes that the user can set through the `OptionsString` parameter, their default initial values, and the name that represents each attribute in the options string.

Table 6-1. IVI Inherent Attribute Initial Values and Options String Name

| Attribute | Default Initial Value | Options String Name |
|-------------------------|-----------------------|---------------------|
| Range Check | True | RangeCheck |
| Query Instrument Status | False | QueryInstrStatus |
| Cache | True | Cache |
| Simulate | False | Simulate |
| Record Value Coercions | False | RecordCoercions |
| Interchange Check | False | InterchangeCheck |
| Driver Setup | "" an empty string | DriverSetup |

The format of an assignment in the `OptionsString` parameter is "`Name=Value`", where `Name` is one of the option string names in the table above. `Initialize` interprets the `Name` and `Value` fields in a case-insensitive manner.

For the attributes of type `ViBoolean`, `Value` can be any of the following:

- To set the attribute to True, use `VI_TRUE`, `True`, or `1`.
- To set the attribute to False, use `VI_FALSE`, `False`, or `0`.

The user can set multiple attributes by separating assignments with commas. If the `OptionsString` parameter contains an assignment for the `DriverSetup` attribute, the `Initialize` function assumes that everything following "`DriverSetup=`" is part of the assignment. Therefore, the user is expected to place the `DriverSetup` assignment at the end of the `OptionsString` parameter. The value that the user passes in the `OptionsString` parameter for the `DriverSetup` attribute must contain only ASCII characters.

Each IVI specific driver defines its own meaning and valid values for the `DriverSetup` attribute. Many specific drivers ignore the value of the `DriverSetup` attribute. Other specific drivers use the `DriverSetup` string to configure instrument specific features at initialization. For example, if a specific driver supports a family of instrument models, the driver can use the `DriverSetup` attribute to allow the user to specify a particular instrument model to simulate.

The IVI specific driver ignores all white space in the `OptionsString` parameter outside the `DriverSetup` string.

If the user attempts to initialize the instrument a second time without first calling the `Close` function, the behavior of the `Initialize` function depends on whether the user accesses the instrument driver through a COM or a C interface.

- If the user accesses the IVI driver through a COM interface and attempts to initialize the instrument a second time without first calling the `Close` function, the `Initialize` function returns the `Already Initialized` error.
- If the user accesses the IVI driver through a C interface and attempts to initialize the instrument a second time without first calling the `Close` function, the `Initialize` function performs all operations that this section defines and returns a new IVI session.

COM Method Prototype

```
HRESULT Initialize([in] BSTR ResourceName,
                  [in] VARIANT_BOOL IdQuery,
                  [in] VARIANT_BOOL Reset,
                  [in,optional] BSTR OptionString);
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_InitWithOptions (ViRsrc ResourceName,
                                          ViBoolean IdQuery,
                                          ViBoolean Reset,
                                          ViConstString OptionString,
                                          ViSession *Vi);

ViStatus _VI_FUNC Prefix_init (ViRsrc ResourceName,
                               ViBoolean IdQuery,
                               ViBoolean Reset,
                               ViSession *Vi);
```

Note: *Prefix_init* exists for compatibility with *VXIplug&play*. Calling *Prefix_init* is equivalent to calling *Prefix_InitWithOptions* with *VI_NULL* or an empty string for the *OptionString* parameter.

Parameters

| Inputs | Description | Data Type |
|--------------|--|---------------|
| ResourceName | An IVI logical name or an instrument specific string that identifies the address of the instrument, such as a VISA resource descriptor string. | ViRsrc |
| IdQuery | Specifies whether to verify the ID of the instrument. | ViBoolean |
| Reset | Specifies whether to reset the instrument. | ViBoolean |
| OptionString | A string that allows the user to specify the initial values of certain inherent attributes. | ViConstString |

| Outputs | Description | Data Type |
|---------|---------------------------------------|-----------|
| Vi | Unique identifier for an IVI session. | ViSession |

Return Values

The table below defines specific status codes that this function returns. Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

| Name | COM Identifier | C Identifier |
|------------------------|----------------------------|--------------------------------|
| ID Query Not Supported | S_IVI_WARN_NSUP_ID_QUERY | IVI_WARN_NSUP_ID_QUERY |
| Reset Not Supported | S_IVI_WARN_NSUP_RESET | IVI_WARN_NSUP_RESET |
| ID Query Failed | E_IVI_ID_QUERY_FAILED | IVI_ERROR_ID_QUERY_FAILED |
| Resource Unknown | E_IVI_RESOURCE_UNKNOWN | IVI_ERROR_RESOURCE_UNKNOWN |
| Missing Option Name | E_IVI_MISSING_OPTION_NAME | IVI_ERROR_MISSING_OPTION_NAME |
| Missing Option Value | E_IVI_MISSING_OPTION_VALUE | IVI_ERROR_MISSING_OPTION_VALUE |
| Bad Option Name | E_IVI_BAD_OPTION_NAME | IVI_ERROR_BAD_OPTION_NAME |
| Bad Option Value | E_IVI_BAD_OPTION_VALUE | IVI_ERROR_BAD_OPTION_VALUE |

Compliance Notes

1. IVI class drivers shall accept only IVI logical names as valid values for the `ResourceName` parameter.

6.15 Invalidate All Attributes

Description

This function invalidates the cached values of all attributes for the session.

COM Method Prototype

```
HRESULT DriverOperation.InvalidateAllAttributes();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_InvalidateAllAttributes (ViSession Vi);
```

Parameters

| Inputs | Description | Data Type |
|--------|---------------------------------------|-----------|
| vi | Unique identifier for an IVI session. | ViSession |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. If the IVI specific driver does not implement state caching, this function shall perform no operations and return Success.

6.16 Lock Session

Description

For IVI-COM specific drivers, this function shall return the Function Not Supported error when the user calls this function. Because of the way thread ownership inherently works in COM, a COM object cannot reliably establish ownership of a lock by relying on the identity of the calling thread. Thus, it is impossible to associate the lock with any single thread. Since this function is already included in the `IviDriver.idl` and removing it from `IviDriver.idl` introduces new versioning and compatibility issues, the specified behavior of this function was changed to return Function Not Supported instead of removing the function from the driver implementation.

For IVI-C specific drivers, this function obtains a multithread lock on the instrument session. Before it does so, Lock Session waits until all other execution threads have released their locks on the instrument session. This capability is useful for developing programs that share the same session among multiple threads.

The user can use Lock Session with IVI-C specific drivers to protect a section of code that requires exclusive access to the instrument. This occurs when the user takes multiple actions that affect the instrument and the user wants to ensure that other execution threads do not disturb the instrument state until all the actions execute. For example, if the user sets various instrument attributes and then triggers a measurement, the user must ensure no other execution thread modifies the attribute values until the user finishes taking the measurement.

With IVI-C drivers, the user can safely make nested calls to Lock Session within the same thread. To completely unlock the session, the user must balance each call to Lock Session with a call to Unlock Session.

The C function has an additional parameter, `CallerHasLock`. If the user uses the `CallerHasLock` parameter in all calls to Lock Session and Unlock Session within a function, the session is locked only once within the function regardless of the number of calls to Lock Session. This allows the user to call Unlock Session just once at the end of the function.

The `CallerHasLock` parameter is useful in complex functions to keep track of whether the user has obtained a lock and therefore needs to unlock the session. The user passes the address of a local variable for the `CallerHasLock` parameter. The user initializes the local variable to `False` in the declaration of the local variable. The user passes the address of the local variable to all other calls to Lock Session or Unlock Session in the same function. Lock Session and Unlock Session each inspect the current value of `CallerHasLock` and take the following actions:

- If the value is `True`, Lock Session does not lock the session again. If the value is `False`, Lock Session obtains the lock and sets the value of the parameter to `True`.
- If the value is `False`, Unlock Session does not attempt to unlock the session. If the value is `True`, Unlock Session releases the lock and sets the value of the parameter to `False`.

If the user passes `VI_NULL` as the `CallerHasLock` parameter from the C interface of the IVI driver, the driver ignores the `CallerHasLock` parameter.

COM Method Prototype

```
HRESULT Utility.LockObject ();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_LockSession (ViSession Vi, ViBoolean *CallerHasLock);
```

Parameters

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

| Input/Output | Description | Data Type |
|---------------|---|-----------|
| CallerHasLock | Indicates whether the calling function currently has a lock on the IVI session. | ViBoolean |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

6.17 Reset

Description

This function performs the following actions:

- Places the instrument in a known state. In an IEEE 488.2 instrument, the Reset function sends the command string "*RST" to the instrument.
- Configures instrument options on which the IVI specific driver depends. A specific driver might enable or disable headers or enable binary mode for waveform transfers.

The user can either call the Reset function separately or specify that it be called from the Initialize function. The Initialize function performs additional operations after performing the reset operation to place the instrument in a state more suitable for interchangeable programming. To reset the device and perform these additional operations, call the Reset With Defaults function instead of the Reset function.

COM Method Prototype

```
HRESULT Utility.Reset();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_reset (ViSession Vi);
```

Parameters

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

Return Values

The table below defines specific status codes that this function returns. Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

| Name | COM Identifier | C Identifier |
|---------------------|------------------|---------------------|
| Reset Not Supported | S_IVI_NSUP_RESET | IVI_WARN_NSUP_RESET |

Compliance Notes

1. If an IVI specific driver performs interchangeability checking, the specific driver shall record an interchangeability warning when the user calls the Reset function.

6.18 Reset Interchange Check

Description

This function resets the interchangeability checking algorithms of the IVI specific driver so that specific driver functions that execute prior to calling this function have no effect on whether future calls to the specific driver generate interchangeability warnings.

When developing a complex test system that consists of multiple test modules, it is generally a good idea to design the test modules so that they can run in any order. To do so requires ensuring that each test module completely configures the state of each instrument it uses. If a particular test module does not completely configure the state of an instrument, the state of the instrument depends on the configuration from a previously executed test module. If the test modules execute in a different order, the behavior of the instrument and therefore the entire test module is likely to change. This change in behavior is generally instrument specific and represents an interchangeability problem.

Users can use this function to test for such cases. By calling this function at the beginning of a test module, users can determine whether the test module has dependencies on the operation of previously executed test modules. Any interchangeability warnings that occur after the user calls this function indicate that the section of the test program that executes after this function and prior to the generation of the warning does not completely configure the instrument and that the user is likely to experience different behavior if the user changes the execution order of the test modules or if the user changes instruments.

Note: This function does not clear interchangeability warnings from the list of interchangeability warnings. To guarantee that the Get Next Interchange Warning function returns interchangeability warnings that occur only *after* the program calls function, the user must clear the list of interchangeability warnings by calling the Clear Interchange Warnings function.

Refer to the Interchange Check attribute for more information on interchangeability checking.

COM Method Prototype

```
HRESULT DriverOperation.ResetInterchangeCheck();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_ResetInterchangeCheck (ViSession Vi);
```

Parameters

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

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. If an IVI-COM specific driver does not accept True as a valid value for the Interchange Check attribute, then the IVI-COM specific driver shall return the Function Not Supported error when the user calls this function.
2. If an IVI-C specific driver does not accept True as a valid value for the Interchange Check attribute, then the IVI-C specific driver shall not export this function.

6.19 Reset With Defaults

Description

The Reset With Defaults function performs the same operations that the Reset function performs and then performs the following additional operations in the specified order:

- Disables the class extension capability groups that the IVI specific driver implements.
- If the class specification with which the IVI specific driver is compliant defines initial values for attributes, this function sets those attributes to the initial values that the class specification defines.
- Configures the initial settings for the specific driver and instrument based on the information retrieved from the IVI configuration store when the instrument driver session was initialized.

Notice that the Initialize function also performs these functions. To place the instrument and the IVI specific driver in the exact same state that they attain when the user calls the Initialize function, the user must first call the Close function and then the Initialize function.

COM Method Prototype

```
HRESULT Utility.ResetWithDefaults ();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_ResetWithDefaults (ViSession Vi);
```

Parameters

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

Return Values

The table below defines specific status codes that this function returns. Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

| Name | COM Identifier | C Identifier |
|---------------------|------------------|---------------------|
| Reset Not Supported | S_IVI_NSUP_RESET | IVI_WARN_NSUP_RESET |

6.20 Revision Query (IVI-C Only)

Description

Obtains the following information:

- The revision of the IVI specific driver
- The firmware revision of the instrument

When calling the Revision Query function through a C interface, the user should pass a buffer with at least 256 bytes for the `DriverRev` parameter.

When calling the Revision Query function through a C interface, the user should pass a buffer with at least 256 bytes for the `InstrRev` parameter.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_revision_query (ViSession Vi,
                                         ViChar DriverRev[],
                                         ViChar InstrRev[])
```

Parameters

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

| Outputs | Description | Data Type |
|-----------|---|-----------|
| DriverRev | Returns the revision of the IVI specific driver, which is the value held in the Specific Driver Revision attribute. Refer to the Specific Driver Revision attribute for more information. | ViChar[] |
| InstrRev | Returns the firmware revision of the instrument, which is the value held in the Instrument Firmware Revision attribute. Refer to the Instrument Firmware Revision attribute for more information. | ViChar[] |

Return Values

The table below defines specific status codes that this function returns. Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

| Name | COM Identifier | C Identifier |
|------------------------------|---------------------------|-------------------------------|
| Revision Query Not Supported | S_IVI_NSUP_REV_QUERY | IVI_WARN_NSUP_REV_QUERY |
| Unexpected Response | E_IVI_UNEXPECTED_RESPONSE | IVI_ERROR_UNEXPECTED_RESPONSE |

Compliance Notes

1. IVI-C specific drivers shall not write more than 256 characters, including the NUL byte, into the `DriverRev` output parameter.
2. IVI-C specific drivers shall not write more than 256 characters, including the NUL byte, into the `InstrRev` output parameter.

6.21 Self Test

Description

Causes the instrument to perform a self test. Self Test waits for the instrument to complete the test. It then queries the instrument for the results of the self test and returns the results to the user.

If the instrument passes the self test, this function returns zero in the `TestResult` parameter and “Self test passed” in the `TestMessage` parameter.

When calling the Self Test function through a C interface, the user should pass a buffer with at least 256 bytes for the `TestMessage` parameter.

COM Method Prototype

```
HRESULT Utility.SelfTest([in,out] long* TestResult,  
                        [in,out] BSTR* TestMessage);
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_self_test(ViSession Vi,  
                                   ViInt16 * TestResult,  
                                   ViChar TestMessage[]);
```

Parameters

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

| Outputs | Description | Data Type |
|-------------|---|-----------|
| TestResult | Returns the numeric result from the self test operation 0 = no error (test passed) | ViInt16 |
| TestMessage | Returns the self test status message | ViChar[] |

Return Values

The table below defines specific status codes that this function returns. Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

| Name | COM Identifier | C Identifier |
|-------------------------|---------------------------|-------------------------------|
| Self Test Not Supported | S_IVI_NSUP_SELF_TEST | IVI_WARN_NSUP_SELF_TEST |
| Unexpected Response | E_IVI_UNEXPECTED_RESPONSE | IVI_ERROR_UNEXPECTED_RESPONSE |

Compliance Notes

1. If the instrument does not return a self test status message, the IVI specific driver shall create and return a message that corresponds to the numeric result that the specific driver returns in the `TestResult` parameter.
2. IVI-C specific drivers shall not write more than 256 characters, including the NUL byte, into the `TestMessage` output parameter.

6.22 Set Attribute <type> (IVI-C Only)

Set Attribute ViInt32
Set Attribute ViInt64
Set Attribute ViReal64
Set Attribute ViString
Set Attribute ViBoolean
Set Attribute ViSession

Description

Sets an attribute to a value. A separate typesafe function exists for each possible attribute data type.

Note: A specific driver may omit the ViInt64 function if the driver has no 64-bit attributes.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_SetAttributeViInt32 (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViInt32 AttributeValue);  
  
ViStatus _VI_FUNC Prefix_SetAttributeViInt64 (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViInt64 AttributeValue);  
  
ViStatus _VI_FUNC Prefix_SetAttributeViReal64 (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViReal64 AttributeValue);  
  
ViStatus _VI_FUNC Prefix_SetAttributeViBoolean (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViBoolean AttributeValue);  
  
ViStatus _VI_FUNC Prefix_SetAttributeViString (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViConstString AttributeValue);  
  
ViStatus _VI_FUNC Prefix_SetAttributeViSession (ViSession Vi,  
                                              ViConstString RepCapIdentifier,  
                                              ViAttr AttributeID,  
                                              ViSession AttributeValue);
```

Parameters

| Inputs | Description | Data Type |
|------------------|--|---|
| Vi | Unique identifier for an IVI session. | ViSession |
| RepCapIdentifier | If the attribute applies to a repeated capability, the user passes a virtual or physical repeated capability identifier. | ViConstString |
| AttributeID | The ID of the attribute. | ViAttr |
| AttributeValue | The value to which to set the attribute. | depends on the data type of the attribute |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

6.23 Unlock Session

Description

For IVI-COM specific drivers, this function shall return the Function Not Supported error when the user calls this function.

For IVI-C specific drivers, this function releases a lock that the Lock Session function acquires. Refer to Lock Session for additional information on IVI session locks.

COM Method Prototype

```
HRESULT Utility.UnlockObject ();
```

C Function Prototype

```
ViStatus _VI_FUNC Prefix_UnlockSession (ViSession Vi, ViBoolean  
*CallerHasLock);
```

Parameters

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

| Input/Output | Description | Data Type |
|---------------|---|-----------|
| CallerHasLock | Indicates if the calling function currently has a lock on the IVI session. Refer to function description for Lock Session for more information. | ViBoolean |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

7. Specific Driver Wrapper Functions

This section defines additional IVI inherent capabilities that IVI-COM and IVI-C specific driver wrappers are required to implement. An IVI specific driver wrapper works with a particular IVI specific driver. An IVI specific driver wrapper provides an interface type that is different from the native interface type of the specific driver. For example, if the native interface type of a specific driver is COM, the specific driver developer can create a wrapper that gives the specific driver a C interface, or vice versa. Wrappers allow specific driver developers to place the majority of the instrument control code in the native driver using one interface type and then build a relatively small wrapper that presents another type of IVI driver interface. The specific driver developer can choose whether the native driver interface is COM or C and then provide a wrapper that presents the alternate interface, C or COM.

The additional capabilities that this section defines are not intended for users and are typically used only by IVI class drivers. Therefore, the additional functions in the C wrappers for IVI-COM drivers should not appear in the function panel file or help information for the specific driver wrapper.

When used in conjunction with an IVI class driver, IVI specific drivers that have wrappers with the additional capabilities that this section defines enable the following scenarios:

- A user calling an IVI class driver through a C interface can call the underlying IVI specific driver through a C interface regardless of whether the C interface is the native interface of the specific driver and regardless of whether the class driver calls the specific driver's COM interface or C interface.
- A user calling an IVI class driver through a COM interface can call the underlying IVI specific driver through a COM interface regardless of whether the COM interface is the native interface of the specific driver and regardless of whether the class driver calls the specific driver's COM interface or C interface.

This section defines a property and a method for COM wrappers and two functions for C wrappers that enable these two scenarios.

C wrappers for IVI-COM specific drivers export the following functions:

- *Prefix_GetNativeIUnknownPtr*
- *Prefix_AttachToExistingCOMSession*

COM wrappers for IVI-C specific drivers export the following property and method:

- *NativeCHandle*
- *AttachToExistingCSession*

7.1 C Wrappers for IVI-COM Specific Drivers

This section defines additional functions that C wrappers for IVI-COM specific drivers are required to export.

7.1.1 Get Native IUnknown Pointer (IVI-C Only)

This function returns the IUnknown interface pointer that the C wrapper is currently using to communicate with an IVI-COM specific driver.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_GetNativeIUnknownPtr (ViSession Vi,  
                                               IUnknown **NativeIUnknownPtr);
```

Parameters

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

| Outputs | Description | Data Type |
|-------------------|--|------------|
| NativeIUnknownPtr | The IUnknown interface pointer that the C wrapper is currently using to communicate with an IVI-COM specific driver. | IUnknown * |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. Only C wrappers for native IVI-COM specific drivers export this function.

7.1.2 Attach To Existing COM Session (IVI-C Only)

Description

This function creates and returns a C wrapper session that can be used to communicate with an existing IVI-COM specific driver session.

COM Method Prototype

N/A

C Function Prototype

```
ViStatus _VI_FUNC Prefix_AttachToExistingCOMSession (IUnknown  
                                                    *ExistingIUnknownPtr,  
                                                    ViSession *Vi);
```

Parameters

| Inputs | Description | Data Type |
|---------------------|---|------------|
| ExistingIUnknownPtr | The IUnknown pointer that corresponds to an existing IVI-COM specific driver session. | IUnknown * |

| Outputs | Description | Data Type |
|---------|---|-----------|
| Vi | The C wrapper session that can be used to communicate with an existing IVI-COM specific driver session. | ViSession |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. Only C wrappers for native IVI-COM specific drivers export this function.

7.2 COM Wrappers for IVI-C Specific Drivers

This section defines an additional interface that COM wrappers for IVI-C specific drivers are required to implement. The name of the additional interface is `IIVIClassWrapper`. The `IIVIClassWrapper` interface is reachable only through `QueryInterface` and contains a property named `NativeCHandle` and a method named `AttachToExistingCSession`.

The following table lists the GUID for the `IIVIClassWrapper` interface.

| Interface | GUID |
|-------------------------------|--|
| <code>IIVIClassWrapper</code> | {47ed518a-a398-11d4-ba58-000064657374} |

The following table shows the property and method of the `IIVIClassWrapper` interface. The Generic Name column lists the generic name for the property and method. The Type column uses a “P” or an “M” to specify whether the item is a property or method.

| COM Interface Hierarchy | Generic Name | Type |
|---------------------------------------|------------------------------|------|
| <code>AttachToExistingCSession</code> | Attach To Existing C Session | M |
| <code>NativeCHandle</code> | Native C Handle | P |

7.2.1 Native C Handle (IVI-COM Only)

| Data Type | Access |
|------------------------|--------|
| <code>ViSession</code> | RO |

COM Property Name

`NativeCHandle`

C Constant Name

N/A

Description

This property returns the C session handle that the COM wrapper is currently using to communicate with an IVI-C specific driver.

Compliance Notes

1. Only COM wrappers for native IVI-C specific drivers export this property.

7.2.2 Attach To Existing C Session (IVI-COM Only)

Description

This method binds a COM wrapper object to an existing IVI-C specific driver session.

COM Method Prototype

```
HRESULT AttachToExistingCSession ([in] long Vi);
```

C Function Prototype

N/A

Parameters

| Inputs | Description | Data Type |
|--------|---------------------------------------|-----------|
| Vi | Unique identifier for an IVI session. | long |

Return Values

Section 9, *Common Error and Completion Codes*, defines general status codes that this function can return.

Compliance Notes

1. Only C wrappers for native IVI-COM specific drivers export this method.

8. IVI Inherent Attribute ID Definitions

This section defines the ID values that IVI-C class drivers and IVI-C specific drivers use for IVI inherent attributes.

Refer to *IVI-3.1: Driver Architecture Specification* for a complete list of the ranges of values that IVI-C drivers use for attribute IDs.

Section 8.1 lists the attribute IDs for the IVI inherent attributes that this specification defines. Sections 8.2, 8.3, and 8.4 list certain values within the `IVI_INHERENT_ATTR_BASE` range that are reserved to retain compatibility with drivers developed before this specification was completed.

8.1 Inherent Attribute ID Values

The following table defines the ID values for the IVI Inherent attributes.

| Attribute Name | ID Value |
|---|---|
| <code>PREFIX_ATTR_RANGE_CHECK</code> | <code>IVI_INHERENT_ATTR_BASE + 2</code> |
| <code>PREFIX_ATTR_QUERY_INSTRUMENT_STATUS</code> | <code>IVI_INHERENT_ATTR_BASE + 3</code> |
| <code>PREFIX_ATTR_CACHE</code> | <code>IVI_INHERENT_ATTR_BASE + 4</code> |
| <code>PREFIX_ATTR_SIMULATE</code> | <code>IVI_INHERENT_ATTR_BASE + 5</code> |
| <code>PREFIX_ATTR_RECORD_COERCIONS</code> | <code>IVI_INHERENT_ATTR_BASE + 6</code> |
| <code>PREFIX_ATTR_DRIVER_SETUP</code> | <code>IVI_INHERENT_ATTR_BASE + 7</code> |
| <code>PREFIX_ATTR_INTERCHANGE_CHECK</code> | <code>IVI_INHERENT_ATTR_BASE + 21</code> |
| <code>PREFIX_ATTR_CLASS_DRIVER_PREFIX</code> | <code>IVI_INHERENT_ATTR_BASE + 301</code> |
| <code>PREFIX_ATTR_SPECIFIC_DRIVER_PREFIX</code> | <code>IVI_INHERENT_ATTR_BASE + 302</code> |
| <code>PREFIX_ATTR_SPECIFIC_DRIVER_LOCATOR</code> | <code>IVI_INHERENT_ATTR_BASE + 303</code> |
| <code>PREFIX_ATTR_IO_RESOURCE_DESCRIPTOR</code> | <code>IVI_INHERENT_ATTR_BASE + 304</code> |
| <code>PREFIX_ATTR_LOGICAL_NAME</code> | <code>IVI_INHERENT_ATTR_BASE + 305</code> |
| <code>PREFIX_ATTR_SUPPORTED_INSTRUMENT_MODELS</code> | <code>IVI_INHERENT_ATTR_BASE + 327</code> |
| <code>PREFIX_ATTR_GROUP_CAPABILITIES</code> | <code>IVI_INHERENT_ATTR_BASE + 401</code> |
| <code>PREFIX_ATTR_INSTRUMENT_FIRMWARE_REVISION</code> | <code>IVI_INHERENT_ATTR_BASE + 510</code> |
| <code>PREFIX_ATTR_INSTRUMENT_MANUFACTURER</code> | <code>IVI_INHERENT_ATTR_BASE + 511</code> |
| <code>PREFIX_ATTR_INSTRUMENT_MODEL</code> | <code>IVI_INHERENT_ATTR_BASE + 512</code> |
| <code>PREFIX_ATTR_SPECIFIC_DRIVER_VENDOR</code> | <code>IVI_INHERENT_ATTR_BASE + 513</code> |
| <code>PREFIX_ATTR_SPECIFIC_DRIVER_DESCRIPTION</code> | <code>IVI_INHERENT_ATTR_BASE + 514</code> |
| <code>PREFIX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION</code> | <code>IVI_INHERENT_ATTR_BASE + 515</code> |
| <code>PREFIX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION</code> | <code>IVI_INHERENT_ATTR_BASE + 516</code> |
| <code>PREFIX_ATTR_CLASS_DRIVER_VENDOR</code> | <code>IVI_INHERENT_ATTR_BASE + 517</code> |
| <code>PREFIX_ATTR_CLASS_DRIVER_DESCRIPTION</code> | <code>IVI_INHERENT_ATTR_BASE + 518</code> |
| <code>PREFIX_ATTR_CLASS_DRIVER_CLASS_SPEC_MAJOR_VERSION</code> | <code>IVI_INHERENT_ATTR_BASE + 519</code> |
| <code>PREFIX_ATTR_CLASS_DRIVER_CLASS_SPEC_MINOR_VERSION</code> | <code>IVI_INHERENT_ATTR_BASE + 520</code> |
| <code>PREFIX_ATTR_SPECIFIC_DRIVER_REVISION</code> | <code>IVI_INHERENT_ATTR_BASE + 551</code> |

| Attribute Name | ID Value |
|-----------------------------------|------------------------------|
| PREFIX_ATTR_CLASS_DRIVER_REVISION | IVI_INHERENT_ATTR_BASE + 552 |

8.2 Reserved Vendor Specific Inherent Extension Attribute ID Values and Constants

The following attribute ID values and C defined constants are reserved for vendor specific inherent attribute extensions. An IVI-C class driver or IVI-C specific driver may export an attribute with one of these ID values only if the driver uses the corresponding C defined constant for the attribute. For vendor specific inherent attribute extensions with C defined constant names that are not listed below, the driver shall use ID values in the range starting at `IVI_VENDOR_INHERENT_EXT_ATTR_BASE`.

| Attribute Name | ID Value |
|--|------------------------------|
| IVI_ATTR_NONE | -1 |
| IVI_ATTR_ALL | IVI_INHERENT_ATTR_BASE + 1 |
| IVI_ATTR_SPY | IVI_INHERENT_ATTR_BASE + 22 |
| IVI_ATTR_USE_SPECIFIC_SIMULATION | IVI_INHERENT_ATTR_BASE + 23 |
| IVI_ATTR_DEFER_UPDATE | IVI_INHERENT_ATTR_BASE + 51 |
| IVI_ATTR_RETURN_DEFERRED_VALUES | IVI_INHERENT_ATTR_BASE + 52 |
| IVI_ATTR_PRIMARY_ERROR | IVI_INHERENT_ATTR_BASE + 101 |
| IVI_ATTR_SECONDARY_ERROR | IVI_INHERENT_ATTR_BASE + 102 |
| IVI_ATTR_ERROR_ELABORATION | IVI_INHERENT_ATTR_BASE + 103 |
| IVI_ATTR_IO_SESSION | IVI_INHERENT_ATTR_BASE + 322 |
| IVI_ATTR_IO_SESSION_TYPE | IVI_INHERENT_ATTR_BASE + 324 |
| IVI_ATTR_FUNCTION_CAPABILITIES | IVI_INHERENT_ATTR_BASE + 402 |
| IVI_ATTR_ATTRIBUTE_CAPABILITIES | IVI_INHERENT_ATTR_BASE + 403 |
| IVI_ATTR_ENGINE_MAJOR_VERSION | IVI_INHERENT_ATTR_BASE + 501 |
| IVI_ATTR_ENGINE_MINOR_VERSION | IVI_INHERENT_ATTR_BASE + 502 |
| IVI_ATTR_SPECIFIC_DRIVER_MAJOR_VERSION | IVI_INHERENT_ATTR_BASE + 503 |
| IVI_ATTR_SPECIFIC_DRIVER_MINOR_VERSION | IVI_INHERENT_ATTR_BASE + 504 |
| IVI_ATTR_CLASS_DRIVER_MAJOR_VERSION | IVI_INHERENT_ATTR_BASE + 505 |
| IVI_ATTR_CLASS_DRIVER_MINOR_VERSION | IVI_INHERENT_ATTR_BASE + 506 |
| IVI_ATTR_ENGINE_REVISION | IVI_INHERENT_ATTR_BASE + 553 |

8.3 Reserved Module Private Attribute ID Values

The following attribute ID values are reserved for module private attributes. IVI software modules can use these attribute ID values only for private or hidden attributes. It is recommended that IVI software modules avoid using these attribute ID values and use the `IVI_MODULE_PRIVATE_ATTR_BASE` to define private attributes instead.

| ID Value |
|---|
| <code>IVI_INHERENT_ATTR_BASE + 321</code> |
| <code>IVI_INHERENT_ATTR_BASE + 601</code> |
| <code>IVI_INHERENT_ATTR_BASE + 602</code> |
| <code>IVI_INHERENT_ATTR_BASE + 603</code> |
| <code>IVI_INHERENT_ATTR_BASE + 704</code> |
| <code>IVI_INHERENT_ATTR_BASE + 708</code> |
| <code>IVI_INHERENT_ATTR_BASE + 801</code> |

8.4 Reserved Standard Cross Class Capabilities Attribute ID Values

The following attribute ID values are reserved for use by *IVI-3.3: Standard Cross Class Capabilities Specification*.

| ID Value |
|---|
| <code>IVI_INHERENT_ATTR_BASE + 203</code> |

9. Common Error and Completion Codes

This section defines the list of IVI error and completion codes. For information on standard error code formats and ranges, refer to *IVI-3.1: Driver Architecture Specification*.

9.1 IVI Error and Completion Codes

The following table lists error and completion codes returned by IVI drivers. The last column provides a generic description for the error.

Refer to *IVI-3.1: Driver Architecture Specification* for a complete list of the base values for the error code bases.

Table 9-1. Error and Completion Codes

| Actual Value | Name | Description String |
|----------------------------|--------------------------|--|
| 0x0 | Success | No message |
| Inherent Error Base + 0x00 | Cannot Recover | Unrecoverable failure |
| Inherent Error Base + 0x01 | Instrument Status | Instrument error detected |
| Inherent Error Base + 0x02 | Cannot Open File | File could not be opened |
| Inherent Error Base + 0x03 | Error Reading File | File is being read |
| Inherent Error Base + 0x04 | Error Writing File | File is being modified |
| Inherent Error Base + 0x0B | Invalid Path Name | The path name is invalid |
| Inherent Error Base + 0x0C | Invalid Attribute | Attribute ID not recognized |
| Inherent Error Base + 0x0D | Attribute Not Writeable | Attribute is read-only |
| Inherent Error Base + 0x0E | Attribute Not Readable | Attribute is write-only |
| Inherent Error Base + 0x10 | Invalid Value | Invalid value for parameter or property |
| Inherent Error Base + 0x11 | Function Not Supported | Function or method not supported |
| Inherent Error Base + 0x12 | Attribute Not Supported | Attribute or property not supported |
| Inherent Error Base + 0x13 | Value Not Supported | The enumeration value for the parameter is not supported |
| Inherent Error Base + 0x15 | Types Do Not Match | The attribute and function parameter types do not match |
| Inherent Error Base + 0x1D | Not Initialized | A connection to the instrument has not been initialized |
| Inherent Error Base + 0x20 | Unknown Channel Name | Channel name specified is not valid for the instrument. |
| Inherent Error Base + 0x23 | Too Many Open Files | Too many files opened |
| Inherent Error Base + 0x44 | Channel Name Required | Channel name required |
| Inherent Error Base + 0x45 | Channel Name Not Allowed | The channel name is not allowed |
| Inherent Error Base + 0x49 | Missing Option Name | The option string contains an entry without a name. |
| Inherent Error Base + 0x4A | Missing Option Value | The option string contains an entry without a value. |
| Inherent Error Base + 0x4B | Bad Option Name | The option string contains an entry with an unknown option name. |

Table 9-1. Error and Completion Codes

| Actual Value | Name | Description String |
|----------------------------|--------------------------------------|--|
| Inherent Error Base + 0x4C | Bad Option Value | The option string contains an entry with an unknown option value. |
| Inherent Error Base + 0x56 | Out of Memory | The necessary memory could not be allocated |
| Inherent Error Base + 0x57 | Operation Pending | Operation in progress |
| Inherent Error Base + 0x58 | Null Pointer | Null pointer passed for parameter or property |
| Inherent Error Base + 0x59 | Unexpected Response | Unexpected response from the instrument |
| Inherent Error Base + 0x5B | File Not Found | File not found |
| Inherent Error Base + 0x5C | Invalid File Format | The file format is invalid |
| Inherent Error Base + 0x5D | Status Not Available | The instrument status is not available |
| Inherent Error Base + 0x5E | ID Query Failed | Instrument ID query failed |
| Inherent Error Base + 0x5F | Reset Failed | Instrument reset failed |
| Inherent Error Base + 0x60 | Resource Unknown | Insufficient location information or resource not present in the system. |
| Inherent Error Base + 0x61 | Already Initialized | The driver is already initialized. |
| Inherent Error Base + 0x62 | Cannot Change Simulation State | The simulation state cannot be changed. |
| Inherent Error Base + 0x63 | Invalid Number of Levels in Selector | Invalid number of levels in selector |
| Inherent Error Base + 0x64 | Invalid Range in Selector | Invalid range in selector |
| Inherent Error Base + 0x65 | Unknown Name in Selector | Unknown name in selector |
| Inherent Error Base + 0x66 | Badly-Formed Selector | Badly-formed selector |
| Inherent Error Base + 0x67 | Unknown Physical Identifier | Unknown physical identifier |
| Inherent Warn Base + 0x65 | ID Query Not Supported | Identification query not supported |
| Inherent Warn Base + 0x66 | Reset Not Supported | Reset operation not supported |
| Inherent Warn Base + 0x67 | Self Test Not Supported | Self test operation not supported |
| Inherent Warn Base + 0x68 | Error Query Not Supported | Error query operation not supported |
| Inherent Warn Base + 0x69 | Revision Query Not Supported | Revision query not supported |

9.2 IVI-C Error and Completion Codes

The following table lists the C Identifiers and the recommended format of the error description string for the Error and Completion Codes defined in Table 9-1.

Note: In the message string column entries listed below, %s is always used to represent the component name that returned the error. Additional format strings parameters are specified as %s1, %s2 etc.

Table 9-2. IVI-C Error and Completion Codes

| Name | C Identifier | C Message String |
|-------------------------|-----------------------------------|--|
| Success | VI_SUCCESS IVI_SUCCESS | No message |
| Cannot Recover | IVI_ERROR_CANNOT_RECOVER | “%s: Failure – cannot recover.” |
| Instrument Status | IVI_ERROR_INSTRUMENT_STATUS | “%s: Instrument error detected. Use ErrorQuery() to determine the error(s).” |
| Cannot Open File | IVI_ERROR_CANNOT_OPEN_FILE | “%s: Cannot open file.” |
| Error Reading File | IVI_ERROR_READING_FILE | “%s: Error reading file.” |
| Error Writing File | IVI_ERROR_WRITING_FILE | “%s: Error writing file.” |
| Invalid Path Name | IVI_ERROR_INVALID_PATHNAME | “%s: The pathname is invalid.” |
| Invalid Attribute | IVI_ERROR_INVALID_ATTRIBUTE | “%s: Attribute ID %s1 not recognized.” <i>%s1 = Attribute ID</i> |
| Attribute Not Writeable | IVI_ERROR_ATTR_NOT_WRITEABLE | “%s: Attribute %s1 is read only.” <i>%s1 = Attribute name</i> |
| Attribute Not Readable | IVI_ERROR_ATTR_NOT_READABLE | “%s: Attribute %s1 is write only.” <i>%s1 = Attribute name</i> |
| Invalid Value | IVI_ERROR_INVALID_VALUE | “%s: Invalid value (%s1) for function %s2, parameter %s3.” <i>%s1 = out-of-range value</i> <i>%s2 = function name</i> <i>%s3 = parameter name</i> |
| Function Not Supported | IVI_ERROR_FUNCTION_NOT_SUPPORTED | “%s: Does not support this class-compliant feature: function %s1.” <i>%s1 = function name</i> |
| Attribute Not Supported | IVI_ERROR_ATTRIBUTE_NOT_SUPPORTED | “%s: Does not support this class-compliant feature: attribute %s1.” <i>%s1 = attribute name</i> |

Table 9-2. IVI-C Error and Completion Codes

| Name | C Identifier | C Message String |
|--------------------------|------------------------------------|---|
| Value Not Supported | IVI_ERROR_VALUE_NOT_SUPPORTED | <p>“%s: Does not support this class-compliant feature: (enumeration) value %s1 passed as the value for parameter %s2 in function %s3.”</p> <p><i>%s1 = enumeration value name or value</i> <i>%s2 = parameter name</i> <i>%s3 = function name</i></p> <p>“%s: Does not support this class-compliant feature: (enumeration) value %s1 passed as the value for attribute %s2.”</p> <p><i>%s1 = enumeration value name or value</i> <i>%s2 = attribute name</i></p> |
| Types Do Not Match | IVI_ERROR_TYPES_DO_NOT_MATCH | <p>“%s: SetAttribute%s1 called for attribute of type %s2.”</p> <p>“%s: GetAttribute%s1 called for attribute of type %s2.”</p> <p><i>%s1 =data type of attribute access function</i> <i>%s2 =data type of attribute</i></p> |
| Not Initialized | IVI_ERROR_NOT_INITIALIZED | “%s: A connection to the instrument has not been established.” |
| Unknown Channel Name | IVI_ERROR_UNKNOWN_CHANNEL_NAME | “%s: Unknown channel name.” |
| Too Many Open Files | IVI_ERROR_TOO_MANY_OPEN_FILES | “%s: Too many files are open.” |
| Channel Name Required | IVI_ERROR_CHANNEL_NAME_REQUIRED | “%s: A channel name is required.” |
| Channel Name Not Allowed | IVI_ERROR_CHANNEL_NAME_NOT_ALLOWED | “%s: The channel name is not allowed.” |
| Missing Option Name | IVI_ERROR_MISSING_OPTION_NAME | “%s: The option string is missing an option name.” |
| Missing Option Value | IVI_ERROR_MISSING_OPTION_VALUE | “%s: The option string is missing an option value.” |
| Bad Option Name | IVI_ERROR_BAD_OPTION_NAME | <p>“%s: The %s1 name in the option string is unknown.”</p> <p><i>%s1 = bad option name</i></p> |
| Bad Option Value | IVI_ERROR_BAD_OPTION_VALUE | <p>“%s: The %s1 value in the option string is unknown.”</p> <p><i>%s1 = bad option value</i></p> |
| Out of Memory | IVI_ERROR_OUT_OF_MEMORY | “%s: Could not allocate necessary memory.” |
| Operation Pending | IVI_ERROR_OPERATION_PENDING | “%s: Operation in progress.” |

Table 9-2. IVI-C Error and Completion Codes

| Name | C Identifier | C Message String |
|--------------------------------------|--|--|
| Null Pointer | IVI_ERROR_NULL_POINTER | “%s: Null pointer passed for function %s1, parameter %s2.” %s1 = function name %s2 = parameter name |
| Unexpected Response | IVI_ERROR_UNEXPECTED_RESPONSE | “%s: Unexpected response from instrument.” |
| File Not Found | IVI_ERROR_FILE_NOT_FOUND | “%s: File not found.” |
| Invalid File Format | IVI_ERROR_INVALID_FILE_FORMAT | “%s: Invalid file format.” |
| Status Not Available | IVI_ERROR_STATUS_NOT_AVAILABLE | “%s: The instrument status is not available.” |
| ID Query Failed | IVI_ERROR_ID_QUERY_FAILED | “%s: Instrument ID query failed.” |
| Reset Failed | IVI_ERROR_RESET_FAILED | “%s: Instrument reset failed.” |
| Resource Unknown | IVI_ERROR_RESOURCE_UNKNOWN | “%s: Unknown resource.” |
| Cannot Change Simulation State | IVI_ERROR_CANNOT_CHANGE_SIMULATION_STATE | “%s: The simulation state cannot be changed.” |
| Invalid Number of Levels in Selector | IVI_ERROR_INVALID_NUMBER_OF_LEVELS_IN_SELECTOR | “%s: The number of levels in the selector is not valid for the %s1 repeated capability.” %s1 = repeated capability name |
| Invalid Range in Selector | IVI_ERROR_INVALID_RANGE_IN_SELECTOR | “%s: The range %s1 is not valid for the repeated capability %s2.” %s1 = range %s2 = repeated capability name |
| Unknown Name in Selector | IVI_ERROR_UNKNOWN_NAME_IN_SELECTOR | “%s: Unknown name in selector.” |
| Badly-Formed Selector | IVI_ERROR_BADLY_FORMED_SELECTOR | “%s: The repeated capability selector is badly-formed.” |
| Unknown Physical Identifier | IVI_ERROR_UNKNOWN_PHYSICAL_IDENTIFIER | “%s: Unknown physical repeated capability selector” |
| ID Query Not Supported | IVI_WARN_NSUP_ID_QUERY | “%s: ID Query is not supported by this instrument.” |
| Reset Not Supported | IVI_WARN_NSUP_RESET | “%s: Reset is not supported by this instrument.” |
| Self Test Not Supported | IVI_WARN_NSUP_SELF_TEST | “%s: Self test is not supported by this instrument.” |
| Error Query Not Supported | IVI_WARN_NSUP_ERROR_QUERY | “%s: Error query is not supported by this instrument.” |
| Revision Query Not Supported | IVI_WARN_NSUP_REV_QUERY | “%s: Firmware revision query is not supported by this instrument.” |

9.3 IVI-COM Error and Completion Codes

The following table specifies the COM Identifiers and the recommended format of the error description string for the Error and Completion Codes defined in Table 9-1.

Note: In the description string table entries listed below, %s is always used to represent the component name. Additional format strings parameters are specified as %s1, %s2 etc.

Table 9-3. IVI-COM Error and Completion Codes

| Name | COM Identifier | COM Message String |
|-------------------------|------------------------------|--|
| Success | S_OK S_IVI_SUCCESS | No message |
| Cannot Recover | E_IVI_CANNOT_RECOVER | “%s: Failure – cannot recover.” |
| Instrument Status | E_IVI_INSTRUMENT_STATUS | “%s: Instrument error detected. Use ErrorQuery() to determine the error(s).” |
| Cannot Open File | E_IVI_CANNOT_OPEN_FILE | “%s: Cannot open file.” |
| Error Reading File | E_IVI_READING_FILE | “%s: Error reading file.” |
| Error Writing File | E_IVI_WRITING_FILE | “%s: Error writing file.” |
| Invalid Path Name | E_IVI_INVALID_PATHNAME | “%s: The pathname is invalid.” |
| Invalid Value | E_IVI_INVALID_VALUE | “%s: Invalid value (%s1) for method %s2, parameter %s3.” <i>%s1 = out-of-range value</i> <i>%s2 = method name</i> <i>%s3 = parameter name</i> |
| Function Not Supported | E_IVI_METHOD_NOT_SUPPORTED | “%s: Does not support this class-compliant feature: method %s1.” <i>%s1 = method name</i> |
| Attribute Not Supported | E_IVI_PROPERTY_NOT_SUPPORTED | “%s: Does not support this class-compliant feature: property %s1.” <i>%s1 = property name</i> |
| Value Not Supported | E_IVI_VALUE_NOT_SUPPORTED | “%s: Does not support this class-compliant feature: (enumeration) value %s1 passed as the value for parameter %s2 in method %s3.” <i>%s1 = enumeration value name or value</i> <i>%s2 = parameter name</i> <i>%s3 = method name</i> “%s: Does not support this class-compliant feature: (enumeration) value %s1 passed as the value for property %s2.” <i>%s1 = enumeration value name or value</i> <i>%s2 = property name</i> |
| Not Initialized | E_IVI_NOT_INITIALIZED | “%s: A connection to the instrument has not been established.” |

Table 9-3. IVI-COM Error and Completion Codes

| Name | COM Identifier | COM Message String |
|--------------------------------------|--|---|
| Unknown Channel Name | E_IVI_UNKNOWN_CHANNEL_NAME | “%s: Unknown channel name. ” |
| Too Many Open Files | E_IVI_TOO_MANY_OPEN_FILES | “%s: Too many files are open. ” |
| Channel Name Required | E_IVI_CHANNEL_NAME_REQUIRED | “%s: A channel name is required. ” |
| Missing Option Name | E_IVI_MISSING_OPTION_NAME | “%s: The option string is missing an option name. ” |
| Missing Option Value | E_IVI_MISSING_OPTION_VALUE | “%s: The option string is missing an option value. ” |
| Bad Option Name | E_IVI_BAD_OPTION_NAME | “%s: The %s1 name in the option string is unknown. ” <i>%s1 = bad option name</i> |
| Bad Option Value | E_IVI_BAD_OPTION_VALUE | “%s: The %s1 value in the option string is unknown. ” <i>%s1 = bad option value</i> |
| Out of Memory | E_IVI_OUT_OF_MEMORY | “%s: Could not allocate necessary memory. ” |
| Operation Pending | E_IVI_OPERATION_PENDING | “%s: Operation in progress. ” |
| Null Pointer | E_IVI_NULL_POINTER | “%s: Null pointer passed for method %s1, parameter %s2. ” <i>%s1 = method name</i> <i>%s2 = parameter name</i> |
| Unexpected Response | E_IVI_UNEXPECTED_RESPONSE | “%s: Unexpected response from instrument. ” |
| File Not Found | E_IVI_FILE_NOT_FOUND | “%s: File not found. ” |
| Invalid File Format | E_IVI_INVALID_FILE_FORMAT | “%s: Invalid file format. ” |
| Status Not Available | E_IVI_STATUS_NOT_AVAILABLE | “%s: The instrument status is not available. ” |
| ID Query Failed | E_IVI_ID_QUERY_FAILED | “%s: Instrument ID query failed. ” |
| Reset Failed | E_IVI_RESET_FAILED | “%s: Instrument reset failed. ” |
| Resource Unknown | E_IVI_RESOURCE_UNKNOWN | “%s: Unknown resource. ” |
| Already Initialized | E_IVI_ALREADY_INITIALIZED | “%s: The driver is already initialized. ” |
| Cannot Change Simulation State | E_IVI_CANNOT_CHANGE_SIMULATION_STATE | “ The simulation state cannot be changed. ” |
| Invalid Number of Levels in Selector | E_IVI_INVALID_NUMBER_OF_LEVELS_IN_SELECTOR | “%s: The number of levels in the selector is not valid for the %s1 repeated capability. ” <i>%s1 = repeated capability name</i> |
| Invalid Range in Selector | E_IVI_INVALID_RANGE_IN_SELECTOR | “%s: The range %s1 is not valid for the repeated capability %s2. ” <i>%s1 = range</i> <i>%s2 = repeated capability name</i> |

Table 9-3. IVI-COM Error and Completion Codes

| Name | COM Identifier | COM Message String |
|------------------------------|-----------------------------------|--|
| Unknown Name in Selector | E_IVI_UNKNOWN_NAME_IN_SELECTOR | “%s: Unknown name in selector.” |
| Badly-Formed Selector | E_IVI_BADLY_FORMED_SELECTOR | “%s: The repeated capability selector is badly-formed.” |
| Unknown Physical Identifier | E_IVI_UNKNOWN_PHYSICAL_IDENTIFIER | “%s: Unknown physical repeated capability selector” |
| ID Query Not Supported | S_IVI_NSUP_ID_QUERY | “%s: ID Query is not supported by this instrument.” |
| Reset Not Supported | S_IVI_NSUP_RESET | “%s: Reset is not supported by this instrument.” |
| Self Test Not Supported | S_IVI_NSUP_SELF_TEST | “%s: Self test is not supported by this instrument.” |
| Error Query Not Supported | S_IVI_NSUP_ERROR_QUERY | “%s: Error query is not supported by this instrument.” |
| Revision Query Not Supported | S_IVI_NSUP_REV_QUERY | “%s: Firmware revision query is not supported by this instrument.” |

9.4 Reserved Vendor Specific Error and Completion Code Values and Constants

The following error and completion code values and C defined constants are reserved for vendor specific error and completion code extensions. An IVI-C class driver or IVI-C specific driver may export an error or completion code with one of these ID values only if the driver uses the corresponding C defined constant for the error or completion code. For vendor specific error and completion code extensions with C defined constant names that are not listed below, the driver shall use ID values in the range starting at IVI_VENDOR_SPECIFIC_ERROR_BASE.

| Error or Completion Code | ID Value |
|-------------------------------------|--------------------------------|
| IVI_ERROR_DRIVER_MODULE_NOT_FOUND | IVI_INHERENT_ERROR_BASE + 0x05 |
| IVI_ERROR_CANNOT_OPEN_DRIVER_MODULE | IVI_INHERENT_ERROR_BASE + 0x06 |
| IVI_ERROR_INVALID_DRIVER_MODULE | IVI_INHERENT_ERROR_BASE + 0x07 |
| IVI_ERROR_UNDEFINED_REFERENCES | IVI_INHERENT_ERROR_BASE + 0x08 |
| IVI_ERROR_FUNCTION_NOT_FOUND | IVI_INHERENT_ERROR_BASE + 0x09 |
| IVI_ERROR_LOADING_DRIVER_MODULE | IVI_INHERENT_ERROR_BASE + 0x0A |
| IVI_ERROR_INVALID_PARAMETER | IVI_INHERENT_ERROR_BASE + 0x0F |
| IVI_ERROR_INVALID_TYPE | IVI_INHERENT_ERROR_BASE + 0x14 |
| IVI_ERROR_MULTIPLE_DEFERRED_SETTING | IVI_INHERENT_ERROR_BASE + 0x16 |
| IVI_ERROR_ITEM_ALREADY_EXISTS | IVI_INHERENT_ERROR_BASE + 0x17 |
| IVI_ERROR_INVALID_CONFIGURATION | IVI_INHERENT_ERROR_BASE + 0x18 |
| IVI_ERROR_VALUE_NOT_AVAILABLE | IVI_INHERENT_ERROR_BASE + 0x19 |
| IVI_ERROR_ATTRIBUTE_VALUE_NOT_KNOWN | IVI_INHERENT_ERROR_BASE + 0x1A |
| IVI_ERROR_NO_RANGE_TABLE | IVI_INHERENT_ERROR_BASE + 0x1B |
| IVI_ERROR_INVALID_RANGE_TABLE | IVI_INHERENT_ERROR_BASE + 0x1C |

| Error or Completion Code | ID Value |
|--|--------------------------------|
| IVI_ERROR_NON_INTERCHANGEABLE_BEHAVIOR | IVI_INHERENT_ERROR_BASE + 0x1E |
| IVI_ERROR_NO_CHANNEL_TABLE | IVI_INHERENT_ERROR_BASE + 0x1F |
| IVI_ERROR_SYS_RSRC_ALLOC | IVI_INHERENT_ERROR_BASE + 0x21 |
| IVI_ERROR_ACCESS_DENIED | IVI_INHERENT_ERROR_BASE + 0x22 |
| IVI_ERROR_UNABLE_TO_CREATE_TEMP_FILE | IVI_INHERENT_ERROR_BASE + 0x24 |
| IVI_ERROR_NO_UNUSED_TEMP_FILENAMES | IVI_INHERENT_ERROR_BASE + 0x25 |
| IVI_ERROR_DISK_FULL | IVI_INHERENT_ERROR_BASE + 0x26 |
| IVI_ERROR_CONFIG_FILE_NOT_FOUND | IVI_INHERENT_ERROR_BASE + 0x27 |
| IVI_ERROR_CANNOT_OPEN_CONFIG_FILE | IVI_INHERENT_ERROR_BASE + 0x28 |
| IVI_ERROR_ERROR_READING_CONFIG_FILE | IVI_INHERENT_ERROR_BASE + 0x29 |
| IVI_ERROR_BAD_INTEGER_IN_CONFIG_FILE | IVI_INHERENT_ERROR_BASE + 0x2A |
| IVI_ERROR_BAD_DOUBLE_IN_CONFIG_FILE | IVI_INHERENT_ERROR_BASE + 0x2B |
| IVI_ERROR_BAD_BOOLEAN_IN_CONFIG_FILE | IVI_INHERENT_ERROR_BASE + 0x2C |
| IVI_ERROR_CONFIG_ENTRY_NOT_FOUND | IVI_INHERENT_ERROR_BASE + 0x2D |
| IVI_ERROR_DRIVER_DLL_INIT_FAILED | IVI_INHERENT_ERROR_BASE + 0x2E |
| IVI_ERROR_DRIVER_UNRESOLVED_SYMBOL | IVI_INHERENT_ERROR_BASE + 0x2F |
| IVI_ERROR_CANNOT_FIND_CVI_RTE | IVI_INHERENT_ERROR_BASE + 0x30 |
| IVI_ERROR_CANNOT_OPEN_CVI_RTE | IVI_INHERENT_ERROR_BASE + 0x31 |
| IVI_ERROR_CVI_RTE_INVALID_FORMAT | IVI_INHERENT_ERROR_BASE + 0x32 |
| IVI_ERROR_CVI_RTE_MISSING_FUNCTION | IVI_INHERENT_ERROR_BASE + 0x33 |
| IVI_ERROR_CVI_RTE_INIT_FAILED | IVI_INHERENT_ERROR_BASE + 0x34 |
| IVI_ERROR_CVI_RTE_UNRESOLVED_SYMBOL | IVI_INHERENT_ERROR_BASE + 0x35 |
| IVI_ERROR_LOADING_CVI_RTE | IVI_INHERENT_ERROR_BASE + 0x36 |
| IVI_ERROR_CANNOT_OPEN_DLL_FOR_EXPORTS | IVI_INHERENT_ERROR_BASE + 0x37 |
| IVI_ERROR_DLL_CORRUPTED | IVI_INHERENT_ERROR_BASE + 0x38 |
| IVI_ERROR_NO_DLL_EXPORT_TABLE | IVI_INHERENT_ERROR_BASE + 0x39 |
| IVI_ERROR_UNKNOWN_DEFAULT_SETUP_ATTR | IVI_INHERENT_ERROR_BASE + 0x3A |
| IVI_ERROR_INVALID_DEFAULT_SETUP_VAL | IVI_INHERENT_ERROR_BASE + 0x3B |
| IVI_ERROR_UNKNOWN_MEMORY_PTR | IVI_INHERENT_ERROR_BASE + 0x3C |
| IVI_ERROR_EMPTY_CHANNEL_LIST | IVI_INHERENT_ERROR_BASE + 0x3D |
| IVI_ERROR_DUPLICATE_CHANNEL_STRING | IVI_INHERENT_ERROR_BASE + 0x3E |
| IVI_ERROR_DUPLICATE_VIRT_CHAN_NAME | IVI_INHERENT_ERROR_BASE + 0x3F |
| IVI_ERROR_MISSING_VIRT_CHAN_NAME | IVI_INHERENT_ERROR_BASE + 0x40 |
| IVI_ERROR_BAD_VIRT_CHAN_NAME | IVI_INHERENT_ERROR_BASE + 0x41 |
| IVI_ERROR_UNASSIGNED_VIRT_CHAN_NAME | IVI_INHERENT_ERROR_BASE + 0x42 |
| IVI_ERROR_BAD_VIRT_CHAN_ASSIGNMENT | IVI_INHERENT_ERROR_BASE + 0x43 |
| IVI_ERROR_ATTR_NOT_VALID_FOR_CHANNEL | IVI_INHERENT_ERROR_BASE + 0x46 |
| IVI_ERROR_ATTR_MUST_BE_CHANNEL_BASED | IVI_INHERENT_ERROR_BASE + 0x47 |
| IVI_ERROR_CHANNEL_ALREADY_EXCLUDED | IVI_INHERENT_ERROR_BASE + 0x48 |
| IVI_ERROR_NOT_CREATED_BY_CLASS | IVI_INHERENT_ERROR_BASE + 0x4D |

| Error or Completion Code | ID Value |
|-------------------------------------|--------------------------------|
| IVI_ERROR_IVI_INI_IS_RESERVED | IVI_INHERENT_ERROR_BASE + 0x4E |
| IVI_ERROR_DUP_RUNTIME_CONFIG_ENTRY | IVI_INHERENT_ERROR_BASE + 0x4F |
| IVI_ERROR_INDEX_IS_ONE_BASED | IVI_INHERENT_ERROR_BASE + 0x50 |
| IVI_ERROR_INDEX_IS_TOO_HIGH | IVI_INHERENT_ERROR_BASE + 0x51 |
| IVI_ERROR_ATTR_NOT_CACHEABLE | IVI_INHERENT_ERROR_BASE + 0x52 |
| IVI_ERROR_ADDR_ATTRS_MUST_BE_HIDDEN | IVI_INHERENT_ERROR_BASE + 0x53 |
| IVI_ERROR_BAD_CHANNEL_NAME | IVI_INHERENT_ERROR_BASE + 0x54 |
| IVI_ERROR_BAD_PREFIX_IN_CONFIG_FILE | IVI_INHERENT_ERROR_BASE + 0x55 |

9.5 Standard COM Error Codes for Use during Driver Development

The following table lists the standard COM error codes that IVI driver developers may use during driver development. It also specifies the recommended format of the error description string for those error codes.

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

Table 9-4. Standard COM Error Codes

| Standard COM Error Code | Description String |
|-------------------------|--------------------------|
| E_ABORT | “%s: Operation aborted.” |
| E_NOTIMPL | “%s: Not implemented.” |

Use E_IVI_METHOD_NOT_SUPPORTED or E_IVI_PROPERTY_NOT_SUPPORTED instead of E_NOTIMPL for methods or properties that the IVI specifications define but that you do not intend to support in the driver. Use E_NOTIMPL for methods or properties you intend to implement but have not yet done so.

9.6 Unused Standard COM Error Codes

The following table lists standard COM error codes that you should avoid using. Instead, use the recommended IVI error codes listed in the second column.

Table 9-5. Recommended IVI Error Codes for Standard COM Errors

| Standard COM Error Code | Recommended IVI Error Code |
|-------------------------|----------------------------|
| E_INVALIDARG | E_IVI_INVALID_VALUE |
| E_OUTOFMEMORY | E_IVI_OUT_OF_MEMORY |
| E_PENDING | E_IVI_OPERATION_PENDING |
| E_POINTER | E_IVI_NULL_POINTER |
| E_UNEXPECTED | E_IVI_CANNOT_RECOVER |

Appendix A. ANSI C Include File

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

```
/*=====*/
/*          I V I          */
/*          */
/* Title:      ivic.h          */
/* Purpose:    Declarations for the Interchangeable Virtual Instruments */
/*             (IVI) Library.  */
/*          */
/*=====*/

#ifndef IVI_HEADER
#define IVI_HEADER

#ifdef __cplusplus
extern "C" {
#endif

/*=====*/
/*= Error constants          */
/*=====*/
#define IVI_INHERENT_ERROR_BASE      (0xBFFA0000)
#define IVI_INHERENT_WARN_BASE      (0x3FFA0000)

/* errors codes*/
#define IVIXXX_ERROR_CANNOT_RECOVER      (IVI_INHERENT_ERROR_BASE + 0x00)
#define IVIXXX_ERROR_INSTRUMENT_STATUS  (IVI_INHERENT_ERROR_BASE + 0x01)
#define IVIXXX_ERROR_CANNOT_OPEN_FILE   (IVI_INHERENT_ERROR_BASE + 0x02)
#define IVIXXX_ERROR_READING_FILE       (IVI_INHERENT_ERROR_BASE + 0x03)
#define IVIXXX_ERROR_WRITING_FILE       (IVI_INHERENT_ERROR_BASE + 0x04)
#define IVIXXX_ERROR_INVALID_PATHNAME   (IVI_INHERENT_ERROR_BASE + 0x0B)
#define IVIXXX_ERROR_INVALID_ATTRIBUTE  (IVI_INHERENT_ERROR_BASE + 0x0C)
#define IVIXXX_ERROR_IVI_ATTR_NOT_WRITEABLE (IVI_INHERENT_ERROR_BASE + 0x0D)
#define IVIXXX_ERROR_ATTR_NOT_READABLE  (IVI_INHERENT_ERROR_BASE + 0x0E)
#define IVIXXX_ERROR_INVALID_VALUE      (IVI_INHERENT_ERROR_BASE + 0x10)
#define IVIXXX_ERROR_FUNCTION_NOT_SUPPORTED (IVI_INHERENT_ERROR_BASE + 0x11)
#define IVIXXX_ERROR_ATTRIBUTE_NOT_SUPPORTED (IVI_INHERENT_ERROR_BASE + 0x12)
#define IVIXXX_ERROR_INVALID_VALUE_NOT_SUPPORTED (IVI_INHERENT_ERROR_BASE + 0x13)
#define IVIXXX_ERROR_TYPES_DO_NOT_MATCH (IVI_INHERENT_ERROR_BASE + 0x15)
#define IVIXXX_ERROR_NOT_INITIALIZED    (IVI_INHERENT_ERROR_BASE + 0x1D)
#define IVIXXX_ERROR_UNKNOWN_CHANNEL_NAME (IVI_INHERENT_ERROR_BASE + 0x20)
#define IVIXXX_ERROR_TOO_MANY_OPEN_FILES (IVI_INHERENT_ERROR_BASE + 0x23)
#define IVIXXX_ERROR_CHANNEL_NAME_REQUIRED (IVI_INHERENT_ERROR_BASE + 0x44)
#define IVIXXX_ERROR_CHANNEL_NAME_NOT_ALLOWED (IVI_INHERENT_ERROR_BASE + 0x45)
#define IVIXXX_ERROR_MISSING_OPTION_NAME (IVI_INHERENT_ERROR_BASE + 0x49)
#define IVIXXX_ERROR_MISSING_OPTION_VALUE (IVI_INHERENT_ERROR_BASE + 0x4A)
#define IVIXXX_ERROR_BAD_OPTION_NAME    (IVI_INHERENT_ERROR_BASE + 0x4B)
#define IVIXXX_ERROR_BAD_OPTION_VALUE   (IVI_INHERENT_ERROR_BASE + 0x4C)
#define IVIXXX_ERROR_OUT_OF_MEMORY      (IVI_INHERENT_ERROR_BASE + 0x56)
#define IVIXXX_ERROR_OPERATION_PENDING  (IVI_INHERENT_ERROR_BASE + 0x57)
#define IVIXXX_ERROR_NULL_POINTER       (IVI_INHERENT_ERROR_BASE + 0x58)
#define IVIXXX_ERROR_UNEXPECTED_RESPONSE (IVI_INHERENT_ERROR_BASE + 0x59)
#define IVIXXX_ERROR_FILE_NOT_FOUND     (IVI_INHERENT_ERROR_BASE + 0x5B)
#define IVIXXX_ERROR_INVALID_FILE_FORMAT (IVI_INHERENT_ERROR_BASE + 0x5C)
#define IVIXXX_ERROR_STATUS_NOT_AVAILABLE (IVI_INHERENT_ERROR_BASE + 0x5D)
#define IVIXXX_ERROR_ID_QUERY_FAILED    (IVI_INHERENT_ERROR_BASE + 0x5E)
#define IVIXXX_ERROR_RESET_FAILED       (IVI_INHERENT_ERROR_BASE + 0x5F)
#define IVIXXX_ERROR_RESOURCE_UNKNOWN   (IVI_INHERENT_ERROR_BASE + 0x60)
#define IVIXXX_ERROR_CANNOT_CHANGE_SIMULATION_STATE (IVI_INHERENT_ERROR_BASE + 0x62)
#define IVIXXX_ERROR_INVALID_NUMBER_OF_LEVELS_IN_SELECTOR (IVI_INHERENT_ERROR_BASE + 0x63)
#define IVIXXX_ERROR_INVALID_RANGE_IN_SELECTOR (IVI_INHERENT_ERROR_BASE + 0x64)
#define IVIXXX_ERROR_UNKNOWN_NAME_IN_SELECTOR (IVI_INHERENT_ERROR_BASE + 0x65)
#define IVIXXX_ERROR_BADLY_FORMED_SELECTOR (IVI_INHERENT_ERROR_BASE + 0x66)
```

```

#define IVIXXX_UNKNOWN_PHYSICAL_IDENTIFIER                (IVI_INHERENT_ERROR_BASE + 0x67)

/* warning codes*/
#define IVIXXX_WARN_NSUP_ID_QUERY                       (IVI_INHERENT_WARN_BASE + 0x65)
#define IVIXXX_WARN_NSUP_RESET                         (IVI_INHERENT_WARN_BASE + 0x66)
#define IVIXXX_WARN_NSUP_SELF_TEST                    (IVI_INHERENT_WARN_BASE + 0x67)
#define IVIXXX_WARN_NSUP_ERROR_QUERY                  (IVI_INHERENT_WARN_BASE + 0x68)
#define IVIXXX_WARN_NSUP_REV_QUERY                    (IVI_INHERENT_WARN_BASE + 0x69)

/*****
/*= Base values for attribute constants.                ===== */
/*= A private attribute is one that is defined for use within  ===== */
/*= that module and is not exported via an include file.    ===== */
/*****
#define IVI_INHERENT_ATTR_BASE                          (1050000)

/*****
/*= IVI Inherent          attributes                    ===== */
/*= The data type of each attribute is listed, followed by the  ===== */
/*= default value or a remark.                               ===== */
/*****

/* defined in IVI-3.2 table 8.1 */

#define IVIXXX_ATTR_RANGE_CHECK                        (IVI_INHERENT_ATTR_BASE + 2)
#define IVIXXX_ATTR_QUERY_INSTRUMENT_STATUS          (IVI_INHERENT_ATTR_BASE + 3)
#define IVIXXX_ATTR_CACHE                            (IVI_INHERENT_ATTR_BASE + 4)
#define IVIXXX_ATTR_SIMULATE                         (IVI_INHERENT_ATTR_BASE + 5)
#define IVIXXX_ATTR_RECORD_COERCIONS                 (IVI_INHERENT_ATTR_BASE + 6)
#define IVIXXX_ATTR_DRIVER_SETUP                     (IVI_INHERENT_ATTR_BASE + 7)
#define IVIXXX_ATTR_INTERCHANGE_CHECK                (IVI_INHERENT_ATTR_BASE + 21)
#define IVIXXX_ATTR_CLASS_DRIVER_PREFIX              (IVI_INHERENT_ATTR_BASE + 301)
#define IVIXXX_ATTR_SPECIFIC_DRIVER_PREFIX           (IVI_INHERENT_ATTR_BASE + 302)
#define IVIXXX_ATTR_SPECIFIC_DRIVER_LOCATOR          (IVI_INHERENT_ATTR_BASE + 303)
#define IVIXXX_ATTR_IO_RESOURCE_DESCRIPTOR           (IVI_INHERENT_ATTR_BASE + 304)
#define IVIXXX_ATTR_LOGICAL_NAME                     (IVI_INHERENT_ATTR_BASE + 305)
#define IVIXXX_ATTR_SUPPORTED_INSTRUMENT_MODELS      (IVI_INHERENT_ATTR_BASE + 327)
#define IVIXXX_ATTR_GROUP_CAPABILITIES                (IVI_INHERENT_ATTR_BASE + 401)
#define IVIXXX_ATTR_INSTRUMENT_FIRMWARE_REVISION    (IVI_INHERENT_ATTR_BASE + 510)
#define IVIXXX_ATTR_INSTRUMENT_MANUFACTURER          (IVI_INHERENT_ATTR_BASE + 511)
#define IVIXXX_ATTR_INSTRUMENT_MODEL                 (IVI_INHERENT_ATTR_BASE + 512)
#define IVIXXX_ATTR_SPECIFIC_DRIVER_VENDOR           (IVI_INHERENT_ATTR_BASE + 513)
#define IVIXXX_ATTR_SPECIFIC_DRIVER_DESCRIPTION      (IVI_INHERENT_ATTR_BASE + 514)
#define IVIXXX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION (IVI_INHERENT_ATTR_BASE + 515)
#define IVIXXX_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION (IVI_INHERENT_ATTR_BASE + 516)
#define IVIXXX_ATTR_CLASS_DRIVER_VENDOR              (IVI_INHERENT_ATTR_BASE + 517)
#define IVIXXX_ATTR_CLASS_DRIVER_DESCRIPTION         (IVI_INHERENT_ATTR_BASE + 518)
#define IVIXXX_ATTR_CLASS_DRIVER_CLASS_SPEC_MAJOR_VERSION (IVI_INHERENT_ATTR_BASE + 519)
#define IVIXXX_ATTR_CLASS_DRIVER_CLASS_SPEC_MINOR_VERSION (IVI_INHERENT_ATTR_BASE + 520)
#define IVIXXX_ATTR_SPECIFIC_DRIVER_REVISION         (IVI_INHERENT_ATTR_BASE + 551)
#define IVIXXX_ATTR_CLASS_DRIVER_REVISION            (IVI_INHERENT_ATTR_BASE + 552)

/*****
*----- Ivi Inherent Instrument Driver Function Declarations -----*
*****/

ViStatus _VI_FUNC Ivixxx_init (ViRsrc ResourceName,
                               ViBoolean IdQuery,
                               ViBoolean Reset,
                               ViSession *Vi);

ViStatus _VI_FUNC Ivixxx_InitWithOptions (ViRsrc ResourceName,
                                           ViBoolean IdQuery,
                                           ViBoolean Reset,
                                           ViConstString OptionsString,
                                           ViSession *Vi);

ViStatus _VI_FUNC Ivixxx_close (ViSession Vi);

/* Attribute Access */

ViStatus _VI_FUNC Ivixxx_SetAttributeViBoolean (ViSession Vi,
                                                ViConstString RepCapIdentifier,

```

```

        ViAttr AttributeID,
        ViBoolean AttributeValue);
ViStatus _VI_FUNC Ivixxx_SetAttributeViInt32 (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViInt32 AttributeValue);
ViStatus _VI_FUNC Ivixxx_SetAttributeViInt64 (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViInt64 AttributeValue);
ViStatus _VI_FUNC Ivixxx_SetAttributeViReal64 (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViReal64 AttributeValue);
ViStatus _VI_FUNC Ivixxx_SetAttributeViSession (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViSession AttributeValue);
ViStatus _VI_FUNC Ivixxx_SetAttributeViString (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViConstString AttributeValue);
ViStatus _VI_FUNC Ivixxx_GetAttributeViBoolean (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViBoolean *AttributeValue);
ViStatus _VI_FUNC Ivixxx_GetAttributeViInt32 (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViInt32 *AttributeValue);
ViStatus _VI_FUNC Ivixxx_GetAttributeViInt64 (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViInt64 *AttributeValue);
ViStatus _VI_FUNC Ivixxx_GetAttributeViReal64 (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViReal64 *AttributeValue);
ViStatus _VI_FUNC Ivixxx_GetAttributeViSession (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViSession *AttributeValue);
ViStatus _VI_FUNC Ivixxx_GetAttributeViString (ViSession Vi,
        ViConstString RepCapIdentifier,
        ViAttr AttributeID,
        ViInt32 AttributeValueBufferSize,
        ViChar AttributeValue[]);

/* Utility */

ViStatus _VI_FUNC Ivixxx_self_test (ViSession Vi,
        ViInt16 *TestResult,
        ViChar TestMessage[]);
ViStatus _VI_FUNC Ivixxx_reset (ViSession Vi);
ViStatus _VI_FUNC Ivixxx_ResetWithDefaults (ViSession Vi);
ViStatus _VI_FUNC Ivixxx_Disable (ViSession Vi);
ViStatus _VI_FUNC Ivixxx_revision_query (ViSession Vi,
        ViChar DriverRevision[],
        ViChar InstrumentRevision[]);
ViStatus _VI_FUNC Ivixxx_error_query (ViSession Vi,
        ViInt32 *ErrorCode,
        ViChar ErrorMessage[]);
ViStatus _VI_FUNC Ivixxx_error_message (ViSession Vi,
        ViStatus ErrorCode,
        ViChar ErrorMessage[]);
ViStatus _VI_FUNC Ivixxx_GetError (ViSession Vi,
        ViStatus *ErrorCode,
        ViInt32 ErrorDescriptionBufferSize,
        ViChar ErrorDescription[]);
ViStatus _VI_FUNC Ivixxx_ClearError (ViSession Vi);
ViStatus _VI_FUNC Ivixxx_LockSession (ViSession Vi,
        ViBoolean *CallerHasLock);

```

```

ViStatus _VI_FUNC Ivixxx_UnlockSession (ViSession Vi,
                                       ViBoolean *CallerHasLock);
ViStatus _VI_FUNC Ivixxx_GetNextCoercionRecord (ViSession Vi,
                                               ViInt32 CoercionRecordBufferSize,
                                               ViChar CoercionRecord[]);
ViStatus _VI_FUNC Ivixxx_GetNextInterchangeWarning (ViSession Vi,
                                                    ViInt32 InterchangeWarningBufferSize,
                                                    ViChar InterchangeWarning[]);
ViStatus _VI_FUNC Ivixxx_ResetInterchangeCheck (ViSession Vi);
ViStatus _VI_FUNC Ivixxx_ClearInterchangeWarnings (ViSession Vi);
ViStatus _VI_FUNC Ivixxx_InvalidAllAttributes (ViSession Vi);

// The following functions should only be implemented with C Class Drivers

ViStatus _VI_FUNC Ivixxx_GetSpecificDriverCHandle (ViSession Vi,
                                                  ViSession *SpecificDriverCHandle);
ViStatus _VI_FUNC Ivixxx_GetSpecificDriverIUnknownPtr (ViSession Vi,
                                                      IUnknown **SpecificDriverIUnknownPtr);

// The following functions should only be implemented with C Wrappers
// over IVI-COM Specific Drivers.

ViStatus _VI_FUNC Ivixxx_GetNativeIUnknownPtr (ViSession Vi,
                                              IUnknown **NativeIUnknownPtr);
ViStatus _VI_FUNC Ivixxx_AttachToExistingCOMSession (IUnknown *ExistingIUnknownPtr,
                                                    ViSession *Vi);

#ifdef __cplusplus
}
#endif

#endif /* IVI_HEADER */

```

Appendix B. COM IDL File

To ease the development of a compliant an IVI-COM driver, the IVI Foundation publishes IDL (Interface Description Language) files that consolidate all the method and property definitions listed in this specification.

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

B.1 *IviDriverTypeLib.idl*

```
if !defined(IVI_DRIVER_TYPELIB_IDL_INCLUDED_)
#define IVI_DRIVER_TYPELIB_IDL_INCLUDED_
/*****
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 * All rights reserved.
 *
 *
 * FILENAME      : IviDriverTypeLib.idl
 *
 * STATUS        : APPROVED.
 * COMPILER      : MSVC++ 6.0, sp4 C++ compiler
 * CONTENT       : IVI Driver (Inherent Features) Standard IDL
 *                type library definition
 *
 * $Archive: /IVI/IviTypeLibraries/IviDriverTypeLib/IviDriverTypeLib.idl $
 * $Revision: 9 $
 *****/

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

[
    uuid(47ed5120-a398-11d4-ba58-000064657374),
    version(1.0),
    helpstring("IviDriver 1.0 Type Library"),
    helpfile("IviDriver.chm")
]
library IviDriverLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

#include "IviDriver.idl"
};

/***** REVISION LOG *****/
* $Log: /IVI/IviTypeLibraries/IviDriverTypeLib/IviDriverTypeLib.idl $
*
* 9    2/19/02 1:53p Jmh00
* JMH Final review prior to vote
*
* 8    11/30/01 3:41p Jmh00
* JMH Changes per comments from Noel 10/30
*
* 7    10/05/01 1:04a Jmh00
* JMH Sent to Glenn Burnside for spec on 10/4/2001
*
* 6    8/27/01 8:25p Jmh00
* JMH RFSigGen changes from Johannes & version fix on IviDriver
*
* 5    8/14/01 10:15p Jmh00
* JMH Restructure IVI type libraries workspace
*
```

```

* 4      8/08/01 3:23p Jmh00
* JMH Final review version
***** END OF FILE *****/
#endif // !defined(IVI_DRIVER_TYPELIB_IDL_INCLUDED_)

```

B.2 IviDriver.idl

```

#if !defined(IVI_BASE_IDL_INCLUDED_)
#define IVI_BASE_IDL_INCLUDED_
/*****
 *
 * (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
 * All rights reserved.
 *
 *
 * FILENAME      : IviDriver.idl
 *
 * STATUS        : APPROVED.
 * COMPILER      : MSVC++ 6.0, sp4 MIDL
 * CONTENT       : IVI Driver (Inherent Capabilities) Standard IDL
 *
 * $Archive: /IVI/IviTypeLibraries/IviDriverTypeLib/IviDriver.idl $
 * $Revision: 7 $
*****/

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

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

#define HELP_DRIVER(x) helpstring(HS_DRIVER_ ## x ## ), helpcontext(HC_DRIVER_ ## x ## )

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

#include "IviDriverEnglish.idl"
#include "winerror.h"

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

interface IiviDriver;
interface IiviDriverUtility;
interface IiviComponentIdentity;
interface IiviDriverIdentity;
interface IiviClassIdentity;
interface IiviDriverOperation;
interface IiviClassWrapper;

#define UUID_IIVI_DRIVER          47ed5184-a398-11d4-ba58-000064657374
#define UUID_IIVI_COMPONENT_IDENTITY 47ed5185-a398-11d4-ba58-000064657374
#define UUID_IIVI_DRIVER_IDENTITY 47ed5186-a398-11d4-ba58-000064657374
#define UUID_IIVI_CLASS_IDENTITY 47ed5187-a398-11d4-ba58-000064657374
#define UUID_IIVI_DRIVER_OPERATION 47ed5188-a398-11d4-ba58-000064657374
#define UUID_IIVI_DRIVER_UTILITY 47ed5189-a398-11d4-ba58-000064657374
#define UUID_IIVI_CLASS_WRAPPER 47ed518a-a398-11d4-ba58-000064657374

//-----
// Enumerations
//-----

```

```

[
    HELP_DRIVER(IVI_DRIVER_HRESULTS)
]
typedef enum IviDriver_ErrorCodes
{
    S_IVI_SUCCESS = S_OK,
    E_IVI_CANNOT_RECOVER = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7000),
    E_IVI_INSTRUMENT_STATUS = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7001),
    E_IVI_CANNOT_OPEN_FILE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7002),
    E_IVI_READING_FILE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7003),
    E_IVI_WRITING_FILE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7004),
    E_IVI_INVALID_PATHNAME = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x700B),
    E_IVI_INVALID_VALUE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7010),
    E_IVI_METHOD_NOT_SUPPORTED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7011),
    E_IVI_PROPERTY_NOT_SUPPORTED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7012),
    E_IVI_VALUE_NOT_SUPPORTED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7013),
    E_IVI_NOT_INITIALIZED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x701D),
    E_IVI_UNKNOWN_CHANNEL_NAME = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7020),
    E_IVI_TOO_MANY_OPEN_FILES = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7023),
    E_IVI_CHANNEL_NAME_REQUIRED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7044),
    E_IVI_MISSING_OPTION_NAME = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7049),
    E_IVI_MISSING_OPTION_VALUE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x704A),
    E_IVI_BAD_OPTION_NAME = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x704B),
    E_IVI_BAD_OPTION_VALUE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x704C),
    E_IVI_OUT_OF_MEMORY = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7056),
    E_IVI_OPERATION_PENDING = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7057),
    E_IVI_NULL_POINTER = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7058),
    E_IVI_UNEXPECTED_RESPONSE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7059),
    E_IVI_FILE_NOT_FOUND = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x705B),
    E_IVI_INVALID_FILE_FORMAT = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x705C),
    E_IVI_STATUS_NOT_AVAILABLE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x705D),
    E_IVI_ID_QUERY_FAILED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x705E),
    E_IVI_RESET_FAILED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x705F),
    E_IVI_RESOURCE_UNKNOWN = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7060),
    E_IVI_ALREADY_INITIALIZED = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7061),
    E_IVI_CANNOT_CHANGE_SIMULATION_STATE = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7062),
    E_IVI_INVALID_NUMBER_OF_LEVELS_IN_SELECTOR = MAKE_HRESULT(SEVERITY_ERROR,
FACILITY_ITF, 0x7063),
    E_IVI_INVALID_RANGE_IN_SELECTOR = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7064),

```

```

    E_IVI_UNKNOWN_NAME_IN_SELECTOR      = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7065),
    E_IVI_BADLY_FORMED_SELECTOR        = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7066),
    E_IVI_UNKNOWN_PHYSICAL_IDENTIFIER  = MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF,
0x7067),
    S_IVI_NSUP_ID_QUERY                = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF,
0x7065),
    S_IVI_NSUP_RESET                   = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF,
0x7066),
    S_IVI_NSUP_SELF_TEST               = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF,
0x7067),
    S_IVI_NSUP_ERROR_QUERY             = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF,
0x7068),
    S_IVI_NSUP_REV_QUERY               = MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF,
0x7069)
} IviDriver_ErrorCodes;

```

```

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

```

```

[
    object,
    uuid(UUID_IIVI_DRIVER),
    HELP_DRIVER(I_IVI_DRIVER),
    oleautomation,
    pointer_default(unique)
]
interface IIVI_Driver : IUnknown
{
//----- DriverOperation Interface Reference
    [ propget, HELP_DRIVER(DRIVER_OPERATION) ]
    HRESULT DriverOperation ([out, retval] IIVI_DriverOperation **pVal);

//----- Identity Interface Reference
    [ propget, HELP_DRIVER(IDENTITY) ]
    HRESULT Identity ([out, retval] IIVI_DriverIdentity **pVal);

//----- Utility Interface Reference
    [ propget, HELP_DRIVER(UTILITY) ]
    HRESULT Utility ([out, retval] IIVI_DriverUtility **pVal);

//----- Initialize
    [ HELP_DRIVER(INITIALIZE) ]
    HRESULT Initialize (
        [in] BSTR ResourceName,
        [in] VARIANT_BOOL IdQuery,
        [in] VARIANT_BOOL Reset,
        [in, optional] BSTR OptionString );

//----- Initialized
    [ propget, HELP_DRIVER(INITIALIZED) ]
    HRESULT Initialized ([out, retval] VARIANT_BOOL *pVal);

//----- Close
    [ HELP_DRIVER(CLOSE) ]
    HRESULT Close ();
};

```

```

//-----
//  Utility Interface
//-----

```

```

[
    object,
    uuid(UUID_IIVI_DRIVER_UTILITY),
    HELP_DRIVER(I_IVI_DRIVER_UTILITY),
    oleautomation,
    pointer_default(unique)
]
interface IIVIUtility : IUnknown
{
//----- Reset
    [ HELP_DRIVER(RESET) ]
    HRESULT Reset ();

//----- ResetWithDefaults
    [ HELP_DRIVER(RESET_WITH_DEFAULTS) ]
    HRESULT ResetWithDefaults ();

//----- Disable
    [ HELP_DRIVER(DISABLE) ]
    HRESULT Disable ();

//----- SelfTest
    [ HELP_DRIVER(SELF_TEST) ]
    HRESULT SelfTest (
        [in,out] long* TestResult,
        [in,out] BSTR* TestMessage);

//----- ErrorQuery
    [ HELP_DRIVER(ERROR_QUERY) ]
    HRESULT ErrorQuery (
        [in,out] long* ErrorCode,
        [in,out] BSTR* ErrorMessage);

//----- LockObject
    [ HELP_DRIVER(LOCK_OBJECT) ]
    HRESULT LockObject ();

//----- UnlockObject
    [ HELP_DRIVER(UNLOCK_OBJECT) ]
    HRESULT UnlockObject ();

};

//-----
// Identity Interfaces
//-----

[
    object,
    uuid(UUID_IIVI_COMPONENT_IDENTITY),
    HELP_DRIVER(I_IVI_COMPONENT_IDENTITY),
    oleautomation,
    pointer_default(unique)
]

interface IIVIComponentIdentity : IUnknown
{
//----- Description
    [ propget, HELP_DRIVER(DESCRIPTION) ]
    HRESULT Description ([out, retval] BSTR *pVal);

//----- Revision
    [ propget, HELP_DRIVER(REVISION) ]

```

```

    HRESULT Revision ([out, retval] BSTR *pVal);

//----- Vendor
    [ propget, HELP_DRIVER(VENDOR) ]
    HRESULT Vendor ([out, retval] BSTR *pVal);
};

[
    object,
    uuid(UUID_IIVI_DRIVER_IDENTITY),
    HELP_DRIVER(I_IVI_DRIVER_IDENTITY),
    oleautomation,
    pointer_default(unique)
]
interface IIVI_DriverIdentity : IIVI_ComponentIdentity
{
//----- InstrumentManufacturer
    [ propget, HELP_DRIVER(INSTRUMENT_MANUFACTURER) ]
    HRESULT InstrumentManufacturer ([out, retval] BSTR *pVal);

//----- InstrumentModel
    [ propget, HELP_DRIVER(INSTRUMENT_MODEL) ]
    HRESULT InstrumentModel ([out, retval] BSTR *pVal);

//----- InstrumentFirmwareRevision
    [ propget, HELP_DRIVER(INSTRUMENT_FIRMWARE_REVISION) ]
    HRESULT InstrumentFirmwareRevision ([out, retval] BSTR *pVal);

//----- Identifier
    [ propget, HELP_DRIVER(IDENTIFIER) ]
    HRESULT Identifier ([out, retval] BSTR *pVal);

//----- SupportedInstrumentModels
    [ propget, HELP_DRIVER(SUPPORTED_INSTRUMENT_MODELS) ]
    HRESULT SupportedInstrumentModels ([out, retval] BSTR *pVal);

//----- SpecificationMajorVersion
    [ propget, HELP_DRIVER(SPECIFICATION_MAJOR_VERSION) ]
    HRESULT SpecificationMajorVersion ([out, retval] long *pVal);

//----- SpecificationMinorVersion
    [ propget, HELP_DRIVER(SPECIFICATION_MINOR_VERSION) ]
    HRESULT SpecificationMinorVersion ([out, retval] long *pVal);

//----- GroupCapabilities
    [ propget, HELP_DRIVER(GROUP_CAPABILITIES) ]
    HRESULT GroupCapabilities ([out, retval] BSTR *pVal);
};

//-----
//  DriverOperation Interface
//-----

[
    object,
    uuid(UUID_IIVI_DRIVER_OPERATION),
    HELP_DRIVER(I_IVI_DRIVER_OPERATION),
    oleautomation,
    pointer_default(unique)
]

```

```

interface IIVI_DriverOperation : IUnknown
{
//----- LogicalName
[ propget, HELP_DRIVER(LOGICAL_NAME) ]
HRESULT LogicalName ([out, retval] BSTR *pVal);

//----- IoResourceDescriptor
[ propget, HELP_DRIVER(IO_RESOURCE_DESCRIPTOR) ]
HRESULT IoResourceDescriptor ([out, retval] BSTR *pVal);

//----- Cache
[ propget, HELP_DRIVER(CACHE) ]
HRESULT Cache ([out, retval] VARIANT_BOOL *pVal);

[ propput, HELP_DRIVER(CACHE) ]
HRESULT Cache ([in] VARIANT_BOOL newVal);

//----- InterchangeCheck
[ propget, HELP_DRIVER(INTERCHANGE_CHECK) ]
HRESULT InterchangeCheck ([out, retval] VARIANT_BOOL *pVal);

[ propput, HELP_DRIVER(INTERCHANGE_CHECK) ]
HRESULT InterchangeCheck ([in] VARIANT_BOOL newVal);

//----- QueryInstrumentStatus
[ propget, HELP_DRIVER(QUERY_INSTRUMENT_STATUS) ]
HRESULT QueryInstrumentStatus ([out, retval] VARIANT_BOOL *pVal);

[ propput, HELP_DRIVER(QUERY_INSTRUMENT_STATUS) ]
HRESULT QueryInstrumentStatus ([in] VARIANT_BOOL newVal);

//----- RangeCheck
[ propget, HELP_DRIVER(RANGE_CHECK) ]
HRESULT RangeCheck ([out, retval] VARIANT_BOOL *pVal);

[ propput, HELP_DRIVER(RANGE_CHECK) ]
HRESULT RangeCheck ([in] VARIANT_BOOL newVal);

//----- RecordCoercions
[ propget, HELP_DRIVER(RECORD_COERCIONS) ]
HRESULT RecordCoercions ([out, retval] VARIANT_BOOL *pVal);

[ propput, HELP_DRIVER(RECORD_COERCIONS) ]
HRESULT RecordCoercions ([in] VARIANT_BOOL newVal);

//----- Simulate
[ propget, HELP_DRIVER(SIMULATE) ]
HRESULT Simulate ([out, retval] VARIANT_BOOL *pVal);

[ propput, HELP_DRIVER(SIMULATE) ]
HRESULT Simulate ([in] VARIANT_BOOL newVal);

//----- DriverSetup
[ propget, HELP_DRIVER(DRIVER_SETUP) ]
HRESULT DriverSetup ([out, retval] BSTR *pVal);

//----- InvalidateAllAttributes
[ HELP_DRIVER(INVALIDATE_ALL_ATTRIBUTES) ]
HRESULT InvalidateAllAttributes ();

//----- ClearInterchangeWarnings
[ HELP_DRIVER(CLEAR_INTERCHANGE_WARNINGS) ]

```

```

HRESULT ClearInterchangeWarnings ();

//----- GetNextInterchangeWarning
[ HELP_DRIVER(GET_NEXT_INTERCHANGE_WARNING) ]
HRESULT GetNextInterchangeWarning (
    [out, retval] BSTR* InterchangeWarning);

//----- ResetInterchangeCheck
[ HELP_DRIVER(RESET_INTERCHANGE_CHECK) ]
HRESULT ResetInterchangeCheck ();

//----- GetNextCoercionRecord
[ HELP_DRIVER(GET_NEXT_COERCION_RECORD) ]
HRESULT GetNextCoercionRecord (
    [out, retval] BSTR* CoercionRecord);

};

//-----
// ClassWrapper Interface
//-----

[
    object,
    uuid(UUID_IIVI_CLASS_WRAPPER),
    HELP_DRIVER(I_IVI_CLASS_WRAPPER),
    oleautomation,
    pointer_default(unique)
]
interface IIVIClassWrapper : IUnknown
{
//----- AttachToExistingCSession
[ HELP_DRIVER(ATTACH_TO_EXISTING_C_SESSION) ]
HRESULT AttachToExistingCSession ([in] long Vi);

//----- NativeCHandle
[ propget, HELP_DRIVER(NATIVE_C_HANDLE) ]
HRESULT NativeCHandle ([out, retval] long *pVal);
};

/***** REVISION LOG *****/
* $Log: /IVI/IviTypeLibraries/IviDriverTypeLib/IviDriver.idl $
*
* 7 2/19/02 1:53p Jmh00
* JMH Final review prior to vote
*
* 6 11/30/01 3:41p Jmh00
* JMH Changes per comments from Noel 10/30
*
* 5 10/05/01 1:04a Jmh00
* JMH Sent to Glenn Burnside for spec on 10/4/2001
*
* 4 8/08/01 3:23p Jmh00
* JMH Final review version
***** END OF FILE *****/
#endif // !defined(IVI_BASE_IDL_INCLUDED_)

```

B.3 IviDriverEnglish.idl

```

#if !defined(IVI_DRIVER_ENGLISH_IDL_INCLUDED_)
#define IVI_DRIVER_ENGLISH_IDL_INCLUDED_
/*****
*
* (C) COPYRIGHT INTERCHANGEABLE VIRTUAL INSTRUMENTS FOUNDATION, 2001,2002
* All rights reserved.

```

```

*
*
* FILENAME      : IviDriverEnglish.idl
*
* STATUS       : APPROVED.
* COMPILER     : MSVC++ 6.0, sp4 MIDL
* CONTENT      : IVI Driver (Inherent Capabilities) Standard IDL
*              help context IDs and help strings
*
* $Archive: /IVI/IviTypeLibraries/IviDriverTypeLib/IviDriverEnglish.idl $
* $Revision: 7 $
*****/

#define HC_DRIVER_BASE 0

//-----
// Enumerations
//-----

#define HC_DRIVER_IVI_DRIVER_HRESULTS          HC_DRIVER_BASE + 0

//-----
// Interface IIVI_DRIVER
//-----

#define HC_DRIVER_I_IVI_DRIVER                HC_DRIVER_BASE + 1

#define HC_DRIVER_DRIVER_OPERATION            HC_DRIVER_BASE + 2
#define HC_DRIVER_IDENTITY                   HC_DRIVER_BASE + 3
#define HC_DRIVER_UTILITY                    HC_DRIVER_BASE + 4
#define HC_DRIVER_INITIALIZE                 HC_DRIVER_BASE + 5
#define HC_DRIVER_INITIALIZED                HC_DRIVER_BASE + 6
#define HC_DRIVER_CLOSE                      HC_DRIVER_BASE + 7

//-----
// Interface IIVI_DRIVER_UTILITY
//-----

#define HC_DRIVER_I_IVI_DRIVER_UTILITY        HC_DRIVER_BASE + 8

#define HC_DRIVER_RESET                      HC_DRIVER_BASE + 9
#define HC_DRIVER_RESET_WITH_DEFAULTS        HC_DRIVER_BASE + 10
#define HC_DRIVER_DISABLE                    HC_DRIVER_BASE + 11
#define HC_DRIVER_SELF_TEST                  HC_DRIVER_BASE + 12
#define HC_DRIVER_ERROR_QUERY                HC_DRIVER_BASE + 13
#define HC_DRIVER_LOCK_OBJECT                HC_DRIVER_BASE + 14
#define HC_DRIVER_UNLOCK_OBJECT              HC_DRIVER_BASE + 15

//-----
// Interface IIVI_IDENTITY_COMPONENT
//-----

#define HC_DRIVER_I_IVI_COMPONENT_IDENTITY    HC_DRIVER_BASE + 16

#define HC_DRIVER_DESCRIPTION                 HC_DRIVER_BASE + 17
#define HC_DRIVER_REVISION                    HC_DRIVER_BASE + 18
#define HC_DRIVER_VENDOR                      HC_DRIVER_BASE + 19

//-----
// Interface IIVI_DRIVER_IDENTITY
//-----

#define HC_DRIVER_I_IVI_DRIVER_IDENTITY       HC_DRIVER_BASE + 20

#define HC_DRIVER_INSTRUMENT_MANUFACTURER     HC_DRIVER_BASE + 21
#define HC_DRIVER_INSTRUMENT_MODEL            HC_DRIVER_BASE + 22

```

```

#define HC_DRIVER_INSTRUMENT_FIRMWARE_REVISION HC_DRIVER_BASE + 23
#define HC_DRIVER_IDENTIFIER HC_DRIVER_BASE + 24
#define HC_DRIVER_SUPPORTED_INSTRUMENT_MODELS HC_DRIVER_BASE + 25
#define HC_DRIVER_SPECIFICATION_MAJOR_VERSION HC_DRIVER_BASE + 26
#define HC_DRIVER_SPECIFICATION_MINOR_VERSION HC_DRIVER_BASE + 27
#define HC_DRIVER_GROUP_CAPABILITIES HC_DRIVER_BASE + 28

//-----
// Interface IIVI_DRIVER_OPERATION
//-----

#define HC_DRIVER_I_IVI_DRIVER_OPERATION HC_DRIVER_BASE + 31

#define HC_DRIVER_LOGICAL_NAME HC_DRIVER_BASE + 32
#define HC_DRIVER_IO_RESOURCE_DESCRIPTOR HC_DRIVER_BASE + 33
#define HC_DRIVER_CACHE HC_DRIVER_BASE + 34
#define HC_DRIVER_INTERCHANGE_CHECK HC_DRIVER_BASE + 35
#define HC_DRIVER_QUERY_INSTRUMENT_STATUS HC_DRIVER_BASE + 36
#define HC_DRIVER_RANGE_CHECK HC_DRIVER_BASE + 37
#define HC_DRIVER_RECORD_COERCIONS HC_DRIVER_BASE + 38
#define HC_DRIVER_SIMULATE HC_DRIVER_BASE + 39
#define HC_DRIVER_DRIVER_SETUP HC_DRIVER_BASE + 40
#define HC_DRIVER_INVALIDATE_ALL_ATTRIBUTES HC_DRIVER_BASE + 41
#define HC_DRIVER_CLEAR_INTERCHANGE_WARNINGS HC_DRIVER_BASE + 42
#define HC_DRIVER_GET_NEXT_INTERCHANGE_WARNING HC_DRIVER_BASE + 43
#define HC_DRIVER_RESET_INTERCHANGE_CHECK HC_DRIVER_BASE + 44
#define HC_DRIVER_GET_NEXT_COERCION_RECORD HC_DRIVER_BASE + 45

//-----
// Interface IIVI_CLASS_WRAPPER
//-----

#define HC_DRIVER_I_IVI_CLASS_WRAPPER HC_DRIVER_BASE + 46

#define HC_DRIVER_ATTACH_TO_EXISTING_C_SESSION HC_DRIVER_BASE + 47
#define HC_DRIVER_NATIVE_C_HANDLE HC_DRIVER_BASE + 48

//-----
// Enumerations
//-----

#define HS_DRIVER_IVI_DRIVER_HRESULTS \
"IVI Driver (inherent features) defined HRESULTS"

//-----
// Interface IIVI_DRIVER
//-----

#define HS_DRIVER_I_IVI_DRIVER \
"IVI Driver root interface"

#define HS_DRIVER_DRIVER_OPERATION \
"Pointer to the IIVI_DRIVER_OPERATION interface"

#define HS_DRIVER_IDENTITY \
"Pointer to the IIVI_DRIVER_IDENTITY interface"

#define HS_DRIVER_UTILITY \
"Pointer to the IIVI_DRIVER_UTILITY interface"

#define HS_DRIVER_INITIALIZE \
"Opens the I/O session to the instrument. Driver methods and properties \
that access the instrument are only accessible after Initialize is called. \
Initialize optionally performs a Reset and queries the instrument to \
validate the instrument model."

#define HS_DRIVER_INITIALIZED \
"Initialized is True between a successful call to the Initialize method and a \

```

```

successful call to the Close method, and False at all other times."

#define HS_DRIVER_CLOSE \
"Closes the I/O session to the instrument.  Driver methods and properties \
that access the instrument are not accessible after Close is called."

//-----
//  Interface IIviDriverUtility
//-----

#define HS_DRIVER_I_IVI_DRIVER_UTILITY \
"IVI Driver utility interface"

#define HS_DRIVER_RESET \
"Places the instrument in a known state and configures instrument options on \
which the IVI specific driver depends (for example, enabling/disabling \
headers).  For an IEEE 488.2 instrument, Reset sends the command string \
*RST to the instrument."

#define HS_DRIVER_RESET_WITH_DEFAULTS \
"Does the equivalent of Reset and then, (1) disables class extension \
capability groups, (2) sets attributes to initial values defined by \
class specs, and (3) configures the driver to option string settings used \
when Initialize was last executed."

#define HS_DRIVER_DISABLE \
"Quickly places the instrument in a state where it has no, or minimal, effect \
on the external system to which it is connected.  This state is not \
necessarily a known state."

#define HS_DRIVER_SELF_TEST \
"Performs an instrument self test, waits for the instrument to complete the \
test, and queries the instrument for the results.  If the instrument passes \
the test, TestResult is zero and TestMessage is 'Self test passed'."

#define HS_DRIVER_ERROR_QUERY \
"Queries the instrument and returns instrument specific error information. \
This function can be used when QueryInstrumentStatus is True to retrieve \
error details when the driver detects an instrument error."

#define HS_DRIVER_LOCK_OBJECT \
"Obtains a multithread lock on the driver after waiting until all other \
execution threads have released their locks on the instrument session."

#define HS_DRIVER_UNLOCK_OBJECT \
"Releases a previously obtained multithread lock."

//-----
//  Interface IIviComponentIdentity
//-----

#define HS_DRIVER_I_IVI_COMPONENT_IDENTITY \
"IVI Driver component identity interface"

#define HS_DRIVER_DESCRIPTION \
"A brief description of the implementing component.  Description is limited to \
256 bytes."

#define HS_DRIVER_REVISION \
"The revision of the implementing component.  Refer to IVI-3.2, Section \
3.1.2.2, for a description of revision syntax and semantics.  Revision is \
limited to 256 bytes."

#define HS_DRIVER_VENDOR \
"The name of the vendor that supplies the implementing component.  Vendor is \
limited to 256 bytes."

//-----
//  Interface IIviDriverIdentity

```

```

//-----
#define HS_DRIVER_I_IVI_DRIVER_IDENTITY \
"IVI Driver driver identity interface"

#define HS_DRIVER_INSTRUMENT_MANUFACTURER \
"The name of the manufacturer reported by the physical instrument. If \
Simulation is enabled or the instrument is not capable of reporting the name \
of the manufacturer, a string is returned that explains the condition. \
Manufacturer is limited to 256 bytes"

#define HS_DRIVER_INSTRUMENT_MODEL \
"The model number or name reported by the physical instrument. If \
Simulation is enabled or the instrument is not capable of reporting the model \
number or name, a string is returned that explains the condition. Model is \
limited to 256 bytes"

#define HS_DRIVER_INSTRUMENT_FIRMWARE_REVISION \
"The firmware revision reported by the physical instrument. If Simulation is \
enabled or the instrument is not capable of reporting the firmware revision, \
a string is returned that explains the condition."

#define HS_DRIVER_IDENTIFIER \
"The case-sensitive unique identifier of the implementing IVI-COM instrument \
driver."

#define HS_DRIVER_SUPPORTED_INSTRUMENT_MODELS \
"A comma-separated list of instrument models that the IVI specific \
driver can control."

#define HS_DRIVER_SPECIFICATION_MAJOR_VERSION \
"For IVI class-compliant drivers, the major version number of the instrument \
class specification. If the driver is not class compliant, the driver \
returns zero."

#define HS_DRIVER_SPECIFICATION_MINOR_VERSION \
"For IVI class-compliant drivers, the minor version number of the instrument \
class specification. If the driver is not class compliant, the driver \
returns zero."

#define HS_DRIVER_GROUP_CAPABILITIES \
"A comma-separated list of the class capability groups implemented by the \
driver. Capability group names are documented in the IVI class \
specifications. If the driver is not class compliant, the driver returns an \
empty string."

//-----
// Interface IIVI_DRIVER_OPERATION
//-----

#define HS_DRIVER_I_IVI_DRIVER_OPERATION \
"IVI Driver operation interface"

#define HS_DRIVER_LOGICAL_NAME \
"Logical Name identifies a driver session in the Configuration Store. If \
Logical Name is not empty, the driver was initialized from information in the \
driver session. If it is empty, the driver was initialized without using the \
Configuration Store."

#define HS_DRIVER_IO_RESOURCE_DESCRIPTOR \
"The resource descriptor specifies the connection to a physical device. It \
is either specified in the Configuration Store or passed in the \
ResourceName parameter of the Initialize function. It is empty if the driver \
is not initialized."

#define HS_DRIVER_CACHE \
"If True, the driver caches instrument settings to avoid unnecessary I/O \
to the instrument. The actual settings that are cached are driver-specific."

#define HS_DRIVER_INTERCHANGE_CHECK \
"If True, the driver maintains a record of interchangeability warnings. \

```

```

If the driver does not support interchangeability checking, attempts to set \
InterchangeCheck to True return an error."

#define HS_DRIVER_QUERY_INSTRUMENT_STATUS \
"If True, the driver queries the instrument status at the end of each method \
or property that performs I/O to the instrument.  If an error is reported, \
use ErrorQuery to retrieve error messages one at a time from the instrument."

#define HS_DRIVER_RANGE_CHECK \
"If True, the driver validates property and parameter values to avoid \
invalid commands to the instrument.  The extent of the validation is \
driver-specific."

#define HS_DRIVER_RECORD_COERCIONS \
"If True, the driver keeps a list of the value coercions it makes for ViInt32 \
and ViReal64 attributes.  If the driver does not support coercion recording, \
attempts to set RecordCoercions to True will return an error."

#define HS_DRIVER_SIMULATE \
"If True, the driver does not perform I/O to the instrument, and returns \
simulated values for output parameters."

#define HS_DRIVER_DRIVER_SETUP \
"The driver setup string.  It is either specified in the Configuration \
Store or passed in the OptionString parameter of the Initialize function. \
Driver setup is empty if the driver is not initialized."

#define HS_DRIVER_INVALIDATE_ALL_ATTRIBUTES \
"Invalidates all of the driver's cached values."

#define HS_DRIVER_CLEAR_INTERCHANGE_WARNINGS \
"Cleares the list of interchangeability warnings that the IVI specific driver \
maintains."

#define HS_DRIVER_GET_NEXT_INTERCHANGE_WARNING \
>Returns the oldest warning from the interchange warning list.  Records are \
only added to the list if InterchangeCheck is True."

#define HS_DRIVER_RESET_INTERCHANGE_CHECK \
"Resets the interchangeability checking algorithms of the driver so that \
methods and properties that executed prior to calling this function have \
no affect on whether future calls to the driver generate \
interchangeability warnings."

#define HS_DRIVER_GET_NEXT_COERCION_RECORD \
>Returns the oldest record from the coercion record list.  Records are only \
added to the list if RecordCoercions is True."

//-----
//  Interface IIviClassWrapper
//-----

#define HS_DRIVER_I_IVI_CLASS_WRAPPER \
"IVI Driver class wrapper interface"

#define HS_DRIVER_ATTACH_TO_EXISTING_C_SESSION \
"Binds a COM wrapper object to an existing IVI-C specific driver session."

#define HS_DRIVER_NATIVE_C_HANDLE \
"The C session handle that the COM wrapper is currently using to communicate \
with an IVI-C specific driver."

/***** REVISION LOG *****/
* $Log: /IVI/IviTypeLibraries/IviDriverTypeLib/IviDriverEnglish.idl $
*
* 7      3/07/02 9:53p Jmh00
* JMH Minor changes to helpstrings
*
* 6      2/19/02 1:53p Jmh00
* JMH Final review prior to vote

```

```
*
* 5      11/30/01 3:41p Jmh00
* JMH Changes per comments from Noel 10/30
*
* 4      10/05/01 1:04a Jmh00
* JMH Sent to Glenn Burnside for spec on 10/4/2001
*
* 3      8/08/01 3:23p Jmh00
* JMH Final review version
***** END OF FILE *****/
#endif // !defined(IVI_DRIVER_ENGLISH_IDL_INCLUDED_)
```