IVI-3.2: Inherent Capabilities Specification

April 15, 2011 Edition
Revision 2.1

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`.

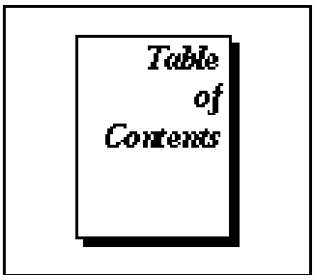
Warranty

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 7

1. Overview of the Inherent Capabilities Specification 9

1.1 Introduction 9
1.2 Inherent Capabilities Overview 9
1.3 References 9
1.4 Definitions of Terms and Acronyms 9

2. Specification Section Layout 10

2.1 Introduction 10
2.1.1 Attribute Section Layout 10
2.1.2 Function Section Layout 10

3. General Requirements 12

3.1 Inherent Capabilities Compliance Rules 12
3.1.1 Attribute Compliance Rules 12
3.1.2 Function Compliance Rules 12
3.1.2.1 Additional Compliance Rules for C Functions with ViChar Array Output Parameters 13
3.1.2.2 Additional Compliance Rules for Revision String Attributes 13
3.1.3 Boolean Attribute and Parameter Values 14
3.2 .NET Namespace 14

4. Inherent Capabilities Overview 15

4.1 .NET Inherent Capabilities 15
4.1.1 Inherent Capabilities Interfaces 16
4.1.1.1 IIviDriver 16
4.1.1.2 IIviDriverOperation 16
4.1.1.3 IIviComponentIdentity 17
4.1.1.4 IIviDriverIdentity 17
4.1.1.5 IIviDriverUtility 17
4.1.1.6 IIviDriverLock 17
4.1.2 Interface Reference Properties 17
4.1.3 IVI.NET IviDriver Session Factory 17
4.2 COM Inherent Capabilities 18
4.2.1 Inherent Capabilities Interfaces 20
4.2.1.1 IIviDriver 20

4.2.1.2 IIviDriverOperation 20
4.2.1.3 IIviComponentIdentity 20
4.2.1.4 IIviDriverIdentity 20
4.2.1.5 IIviDriverUtility 20
4.2.2 Interface Reference Properties 20
4.2.3 IviDriver COM Category 21
4.3 C Inherent Capabilities 22
4.4 Relationship of Inherent Attributes and Different Types of IVI Driver 25

5. Inherent Property/Attribute Descriptions 26
5.1 Cache 26
5.2 Class Driver Class Spec Major Version (IVI-C Only) 27
5.3 Class Driver Class Spec Minor Version (IVI-C Only) 28
5.4 Class Driver Description (IVI-C Only) 29
5.5 Class Driver Prefix (IVI-C Only) 30
5.6 Class Driver Revision (IVI-C Only) 31
5.7 Class Driver Vendor (IVI-C Only) 32
5.8 Class Group Capabilities (IVI-C & IVI-COM Only) 33
5.9 Component Class Spec Major Version (IVI-COM & IVI.NET Only) 34
5.10 Component Class Spec Minor Version (IVI-COM & IVI.NET Only) 35
5.11 Component Description (IVI-COM & IVI.NET Only) 36
5.12 Component Identifier (IVI-COM & IVI.NET Only) 37
5.13 Component Revision (IVI-COM & IVI.NET Only) 38
5.14 Component Vendor (IVI-COM & IVI.NET Only) 39
5.15 Driver Setup 40
5.16 I/O Resource Descriptor 41
5.17 Initialized (IVI-COM Only) 42
5.18 Instrument Firmware Revision 43
5.19 Instrument Manufacturer 44
5.20 Instrument Model 45
5.21 Interchange Check 46
5.22 Logical Name 48
5.23 Query Instrument Status 49
5.24 Range Check 50
5.25 Record Value Coercions 51
5.26 Simulate 52
5.27 Specific Driver Class Spec Major Version (IVI-C Only) 53
5.28 Specific Driver Class Spec Minor Version (IVI-C Only) 54
5.29 Specific Driver Description (IVI-C Only) 55
5.30 Specific Driver Locator (IVI-C Only) 56
5.31 Specific Driver Prefix (IVI-C Only) 57
5.32 Specific Driver Revision (IVI-C Only) 58
5.33 Specific Driver Vendor (IVI-C Only) 59
5.34 Supported Instrument Models (IVI-C & IVI-COM Only) 60

6. Inherent Method/Function Descriptions 61
6.1 Clear Error (IVI-C Only) 61
6.2 Clear Interchange Warnings (IVI-C & IVI-COM Only) 62
6.3 Close 63
6.4 Disable 65
6.5 Error Message (IVI-C Only) 66
6.6 Error Query 67
6.7 Get Attribute <type> (IVI-C Only) 69
6.8 Get Attribute ViString (IVI-C Only) 71

6.9 Get Error (IVI-C Only) 72
6.10 Get Group Capabilities (IVI.NET Only) 74
6.11 Get Next Coercion Record (IVI-C & IVI-COM Only) 75
6.12 Get Next Interchange Warning (IVI-C & IVI-COM Only) 77
6.13 Get Specific Driver C Handle (IVI-C Only) 79
6.14 Get Specific Driver IUnknown Pointer (IVI-C Only) 80
6.15 Get Supported Instrument Models (IVI.NET Only) 81
6.16 Initialize (IVI-C & IVI-COM Only) 82
6.17 Invalidate All Attributes 87
6.18 Lock Session 88
6.19 Reset 91
6.20 Reset Interchange Check 93
6.21 Reset With Defaults 95
6.22 Revision Query (IVI-C Only) 96
6.23 Self Test 98
6.24 Set Attribute <type> (IVI-C Only) 101
6.25 Unlock Session 103

7. Specific Driver Wrapper Functions 104
7.1 C Wrappers for IVI-COM Specific Drivers 105
7.1.1 Get Native IUnknown Pointer (IVI-C Only) 105
7.1.2 Attach To Existing COM Session (IVI-C Only) 106
7.2 IVI-COM and IVI.NET Wrappers for IVI-C Specific Drivers 107
7.2.1 Native C Handle (IVI-COM Only) 107
7.2.2 Attach To Existing C Session (IVI-COM Only) 108

8. IVI.NET Specific Driver Constructor 109

9. IVI.NET Event Descriptions 115
9.1 IVI.NET Events 115
9.1.1 Coercion Record Event (IVI.NET Only) 116
9.1.2 Interchange Check Warning Event (IVI.NET Only) 117
9.1.3 Driver Warning Event (IVI.NET Only) 118

10. IVI Inherent Attribute ID Definitions 119
10.1 Inherent Attribute ID Values 119
10.2 Reserved Vendor Specific Inherent Extension Attribute ID Values and Constants 120
10.3 Reserved Module Private Attribute ID Values 121
10.4 Reserved Standard Cross Class Capabilities Attribute ID Values 121

11. Common IVI-C and IVI-COM Error and Completion Codes 122
11.1 IVI-C and IVI-COM Error and Completion Codes 122
11.2 IVI-C Error and Completion Codes 124
11.3 IVI-COM Error and Completion Codes 127
11.4 Reserved Vendor Specific Error and Completion Code Values and Constants 129
11.5 Standard COM Error Codes for Use during Driver Development 131
11.6 Unused Standard COM Error Codes 131

12. Common IVI.NET Exceptions and Warnings 132

12.1 General Information About Exceptions 132
12.2 Mapping IVI-C and IVI-COM Error Codes to IVI.NET 133
12.3 Mapping IVI-COM Session Factory Error Codes to IVI.NET 134
12.4 Common Exceptions 136
12.4.1 System.ArgumentNullException (.NET Framework) 137
12.4.2 System.InsufficientMemoryException 138
12.4.3 System.IO.FileNotFoundException 139
12.4.4 ConfigurationServerException 140
12.4.5 FileFormatException 141
12.4.6 IdQueryFailedException 142
12.4.7 InstrumentStatusException 143
12.4.8 InvalidOptionValueException 144
12.4.9 IOException 145
12.4.10 IOTimeoutException 146
12.4.11 IviCDriverException 147
12.4.12 IviComDriverException 148
12.4.13 MaxTimeExceededException 149
12.4.14 OperationNotSupportedException 150
12.4.15 OperationPendingException 151
12.4.16 OptionMissingException 152
12.4.17 OptionStringFormatException 153
12.4.18 OutOfRangeException 154
12.4.19 ResetFailedException 155
12.4.20 ResetNotSupportedException 156
12.4.21 SelectorFormatException 157
12.4.22 SelectorHierarchyException 158
12.4.23 SelectorNameException 159
12.4.24 SelectorNameRequiredException 160
12.4.25 SelectorRangeException 161
12.4.26 SimulationStateException 162
12.4.27 TriggerNotSoftwareException 163
12.4.28 UnexpectedResponseException 164
12.4.29 UnknownOptionException 165
12.4.30 UnknownPhysicalNameException 166
12.4.31 ValueNotSupportedException 167
12.5 IVI.NET Session Factory Method Exceptions 168
12.5.1 ClassNameNotFoundException 169
12.5.2 ConfigurationStoreLoadException 171
12.5.3 DriverClassCreationException 172
12.5.4 InvalidClassNameException 173
12.5.5 SessionNotFoundException 174
12.5.6 SoftwareModuleNotFoundException 175
12.6 Warnings 176

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 1.0	April 15, 2002	First approved version. 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: IIviComponentIdentity "Description" property for IVI-COM, CLASS_DRIVER_DESCRIPTION attribute for IVI-C, SPECIFIC_DRIVER_DESCRIPTION attribute for IVI-C. Editorial change to update the IVI Foundation contact information in the Important Information section to remove obsolete address information and refer only to the IVI Foundation web site.
Revision 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.
Revision 2.0	June 9, 2010	Incorporated IVI.NET
Revision 2.1	October 14, 2010	Correct the names of two exceptions.
Revision 2.1	April 15, 2011	Editorial IVI.NET changes. Change the IVI.NET Warning event args to include a text field and add the GUID code parameter name, and add IVI.NET warning messages. Clarify the content of the IVI.NET Coercion Reporting event messages. Clarify the ability to throw derived exceptions from IVI.NET drivers. Remove instrument model information from various exceptions. For IVI.NET exceptions dealing with nested repeated capabilities, clarify the repeated capability to be reported in the exception.

API Versions

Architecture	Drivers that comply with version 2.1 comply with all of the versions below
C	1.0, 1.1, 1.2, 2.0
COM	1.0, 1.1, 1.2, 2.0
.NET	2.0

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

1. Overview of 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.NET, IVI-COM and IVI-C drivers, as well as .NET, COM and C wrappers for native IVI.NET, 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
- IVI-3.17: Installation Specification
- IVI-3.18: .NET Utility Classes and Interfaces
- 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:

DataType—Specifies the VXI*plug&play* data type of the attribute. Valid types are specified in Section 5.14, *Allowed Data Types*, of *IVI-3.1 Driver Architecture Specification*. In some cases where IVI.NET defines a specific type, the .NET type will be used (e.g. Ivi.Driver.PrecisionTimeSpan).

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.

.NET Property Name—Defines the property name, including the object hierarchy, that an IVI.NET specific driver uses for the property. C# syntax is used.

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.

.NET Method Prototype—Defines the method prototype, including the object hierarchy, that an IVI.NET specific driver uses for the method. C# syntax is used.

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 `Prefi`x 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 5.19, *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.NET, IVI-COM, and IVI-C architectures.

Boolean Value	IVI.NET Identifier	IVI-COM Identifier	IVI-C Identifier
True	`true`	`VARIANT_TRUE`	`VI_TRUE`
False	`false`	`VARIANT_FALSE`	`VI_FALSE`

3.2 .NET Namespace

The .NET namespace for the IVI inherent capabilities is `Ivi.Driver`. Note that IVI-3.18, *.NET Utility Classes and Interfaces,* defines additional elements in the `Ivi.Driver` namespace.

4. Inherent Capabilities Overview

This section gives an overview of the inherent capabilities of IVI.NET, IVI-COM, and IVI-C drivers. The inherent capabilities for IVI.NET and 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, COM or .NET properties and methods have corresponding C attributes and functions, and vice versa. 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 .NET Inherent Capabilities

The following table shows the inherent capabilities of an IVI.NET driver. The .NET Interface Hierarchy specifies the relationship of the inherent properties, methods, and events for IVI.NET drivers. The Generic Name column lists the generic name for each property or method. The Type column uses a P, M, or E to specify whether the item is a property, method, or event. IVI.NET is the only IVI API type that defines events.

There is no Initialize() method in the IVI.NET inherent capabilities API, as there is in the IVI-COM and IVI-C APIs. Instead, the IVI.NET specific driver constructor takes the same parameters as Initialize() in IVI-COM, and initializes the driver. The IVI.NET specific driver constructor is described in section 8, *IVI.NET Specific Driver Constructor*.

The IVI.NET inherent capabilities API includes three events, Driver Warning Event, Coercion Record Event, and Interchange Check Warning Event. The IVI.NET events are described in section 9, *IVI.NET Event Descriptions*.

The IVI.NET inherent capabilities do not define Lock and Unlock methods. See section 6.18, *Lock Session*, for details related to the reason that COM does not implement these methods.

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

.NET Interface Hierarchy	Generic Name	Type
`Driver Constructor`	Initialize	M
`Close`	Close	M
`DriverOperation`		
`Cache`	Cache	P
`CoercionRecordEvent`	Coercion Record Event	E
`DriverSetup`	Driver Setup	P
`InterchangeCheck`	Interchange Check	P
`InterchangeCheckWarningEvent`	Interchange Check Warning Event	E
`InvalidateAllAttributes`	Invalidate All Attributes	M
`LogicalName`	Logical Name	P
`QueryInstrumentStatus`	Query Instrument Status	P
`RangeCheck`	Range Check	P
`ResetInterchangeCheck`	Reset Interchange Check	M
`IoResourceDescriptor`	I/O Resource Descriptor	P
`Simulate`	Simulate	P
`WarningEvent`	Driver Warning Event	E

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

.NET Interface Hierarchy	Generic Name	Type
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
SupportedInstrumentModels	Supported Instrument Models	P
GroupCapabilities	Class Group Capabilities	P
Utility		
Disable	Disable	M
ErrorQuery	Error Query	M
Lock	LockSession	M
Reset	Reset	M
ResetWithDefaults	Reset With Defaults	M
SelfTest	Self Test	M
Unlock Note that Unlock is not part of the IVI.NET hierarchy, but is implemented by lock objects associated with the driver.	Unlock Session	M

4.1.1 Inherent Capabilities Interfaces

IVI.NET inherent capabilities are organized into five interfaces.

- IIviDriver
- IIviDriverOperation
- IIviComponentIdentity
- IIviDriverIdentity
- IIviDriverUtility

4.1.1.1 IIviDriver

IIviDriver is the root interface for all IVI.NET drivers. It contains a method that closes the instrument connection. It also contains three interface reference properties. Refer to Section 4.1.2, *Interface Reference Properties,* for more information. IIviDriver derives from IServiceProvider and IDisposable.

4.1.1.2 IIviDriverOperation

IIviDriverOperation contains methods and properties that manage the operation of the driver.

4.1.1.3 IIviComponentIdentity

IIviComponentIdentity contains properties that return general information related to the identity of an IVI component.

4.1.1.4 IIviDriverIdentity

IIviDriverIdentity derives from IIviComponentIdentity. It adds properties that return information related to the identity of the driver and of the instrument.

4.1.1.5 IIviDriverUtility

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

4.1.1.6 IIviDriverLock

The IIviDriverLock interface is returned by calls to the two overloads of the IIviDriverUtility.Lock method. The Lock method is used by a client to obtain a multithread lock for the duration of several method calls. The class implementing IIviDriverLock obtains the lock in its constructor. This blocks the caller of the IIviDriverUtility.Lock method until the lock can be obtained (or the specified timeout period expires).

Once an IIviDriverLock reference is obtained from the Lock method, the client holds the driver lock until the IIviDriverLock.Unlock method is called. IIviDriverLock derives from IDisposable so that classes implementing IIviDriverLock can automatically call Unlock in the Dispose method. This is specifically designed to facilitate the usage of the C# "using" and VB.NET "Using" statements.

4.1.2 Interface Reference Properties

Interface reference properties are used to navigate the .NET Inherent Capabilities hierarchy. Refer to Section 5.17.5, *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 IIviDriver interface defines.

Data Type	Access
`IIviDriverOperation`	DriverOperation
`IIviDriverIdentity`	Identity
`IIviDriverUtility`	Utility

4.1.3 IVI.NET IviDriver Session Factory

The IviDriver .NET assembly contains a factory method called Create for creating instances of generic specific IVI.NET drivers from driver sessions and logical names. Create is a static method accessible from the static IviDriver class.

Refer to *IVI-3.5: Configuration Server Specification* for a description of how logical names and session names are defined in the configuration store.

Refer to Section 8, *IVI.NET Specific Driver Constructor* for more details on how the `idQuery`, `reset`, and `options` parameters affect the instantiation of the driver.

.NET Method Prototype

```
IIviDriver IviDriver.Create(String name,
                            Boolean idQuery,
                            Boolean reset);
IIviDriver IviDriver.Create(String name,
                            Boolean idQuery,
                            Boolean reset,
                            String options);
```

Parameters

Inputs	Description	Base Type
name	A session name or a logical name that points to a session that uses a generic specific IVI.NET driver.	String
idQuery	Specifies whether to verify the ID of the instrument. The default is False.	Boolean
reset	Specifies whether to reset the instrument. The default is False.	Boolean
options	A string that allows the user to specify the initial values of certain inherent attributes. The default is an empty string.	String

Outputs	Description	Base Type
Return Value	Interface pointer to the IIviDriver interface of the driver referenced by session.	IIviDriver

.NET Exceptions

Section **Error! Reference source not found.**, **Error! Reference source not found.**, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Usage

To create a generic specific IVI.NET driver from the logical name "My LogicalName", use the following code:

```
IIviDriver driver = IviDriver.Create("MyLogicalName");
```

In this case, the ID of the instrument will not be verified, the instrument will not be reset, and options will be supplied from the configuration store and/or driver defaults.

4.2 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		

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

COM Interface Hierarchy	Generic Name	Type
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
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

4.2.1 Inherent Capabilities Interfaces

The IIviDriver interface contains interface reference properties for accessing the following inherent capability interfaces:

- IIviDriverOperation
- IIviComponentIdentity
- IIviDriverIdentity
- IIviDriverUtility

Table 4-2 lists the IVI-COM interfaces and their GUIDs.

Table 4-2. Inherent Capabilities COM Interface GUIDs

Interface	GUID
IIviDriver	{47ed5184-a398-11d4-ba58-000064657374}
IIviDriverOperation	{47ed5188-a398-11d4-ba58-000064657374}
IIviComponentIdentity	{47ed5185-a398-11d4-ba58-000064657374}
IIviDriverIdentity	{47ed5186-a398-11d4-ba58-000064657374}
IIviDriverUtility	{47ed5189-a398-11d4-ba58-000064657374}

4.2.1.1 IIviDriver

IIviDriver 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.

4.2.1.2 IIviDriverOperation

IIviDriverOperation contains methods and properties that manage the operation of the driver.

4.2.1.3 IIviComponentIdentity

IIviComponentIdentity contains properties that return general information related to the identity of an IVI component.

4.2.1.4 IIviDriverIdentity

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

4.2.1.5 IIviDriverUtility

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

4.2.2 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 IIviDriver interface defines.

Data Type	Access
IIviDriverOperation	DriverOperation
IIviDriverIdentity	Identity
IIviDriverUtility	Utility

4.2.3 IviDriver COM Category

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

4.3 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
Inherent IVI Attributes		
User Options		
Range Check	*PREFIX*_ATTR_RANGE_CHECK	DriverOperation
Query Instrument Status	*PREFIX*_ATTR_QUERY_INSTRUMENT_STATUS	DriverOperation
Cache	*PREFIX*_ATTR_CACHE	DriverOperation
Simulate	*PREFIX*_ATTR_SIMULATE	DriverOperation
Record Value Coercions	*PREFIX*_ATTR_RECORD_COERCIONS	DriverOperation
Interchange Check	*PREFIX*_ATTR_INTERCHANGE_CHECK	DriverOperation
Class Driver Identification		
Class Driver Description	*PREFIX*_ATTR_CLASS_DRIVER_DESCRIPTION	N/A
Class Driver Prefix	*PREFIX*_ATTR_CLASS_DRIVER_PREFIX	N/A
Class Driver Vendor	*PREFIX*_ATTR_CLASS_DRIVER_VENDOR	N/A
Class Driver Revision	*PREFIX*_ATTR_CLASS_DRIVER_REVISION	N/A
Class Driver Class Spec Major Version	*PREFIX*_ATTR_CLASS_DRIVER_CLASS_SPEC_MAJOR_VERSION	N/A
Class Driver Class Spec Minor Version	*PREFIX*_ATTR_CLASS_DRIVER_CLASS_SPEC_MINOR_VERSION	N/A
Driver Identification		
Specific Driver Description	*PREFIX*_ATTR_SPECIFIC_DRIVER_DESCRIPTION	N/A
Specific Driver Prefix	*PREFIX*_ATTR_SPECIFIC_DRIVER_PREFIX	N/A
Specific Driver Locator	*PREFIX*_ATTR_SPECIFIC_DRIVER_LOCATOR	N/A
Specific Driver Vendor	*PREFIX*_ATTR_SPECIFIC_DRIVER_VENDOR	N/A
Specific Driver Revision	*PREFIX*_ATTR_SPECIFIC_DRIVER_REVISION	N/A
Specific Driver Class Spec Major Version	*PREFIX*_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION	N/A
Specific Driver Class Spec Minor Version	*PREFIX*_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION	N/A
Driver Capabilities		

Table 4-2. Hierarchy of C Inherent Attributes

Category or Generic Attribute Name	C Defined Constant	COM Interface
Supported Instrument Models	*PREFIX*_ATTR_SUPPORTED_INSTRUMENT_MODELS	Identity
Class Group Capabilities	*PREFIX*_ATTR_GROUP_CAPABILITIES	Identity
Instrument Identification		
Instrument Manufacturer	*PREFIX*_ATTR_INSTRUMENT_MANUFACTURER	Identity
Instrument Model	*PREFIX*_ATTR_INSTRUMENT_MODEL	Identity
Instrument Firmware Revision	*PREFIX*_ATTR_INSTRUMENT_FIRMWARE_REVISION	Identity
Advanced Session Information		
Logical Name	*PREFIX*_ATTR_LOGICAL_NAME	DriverOperation
I/O Resource Descriptor	*PREFIX*_ATTR_IO_RESOURCE_DESCRIPTOR	DriverOperation
Driver Setup	*PREFIX*_ATTR_DRIVER_SETUP	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	*Prefix*_init	N/A
Initialize With Options	*Prefix*_InitWithOptions	Main
Attribute Access Functions		
Set Attribute Functions	Prefix_SetAttribute<type>	N/A
Get Attribute Functions	*Prefix*_GetAttribute<type>	N/A
Invalidate All Attributes	*Prefix*_InvalidateAllAttributes	DriverOperation
Utility Functions		
Self Test	*Prefix*_self_test	Utility
Reset	*Prefix*_reset	Utility
ResetWithDefaults	*Prefix*_ResetWithDefaults	Utility
Disable	*Prefix*_Disable	Utility
Revision Query	*Prefix*_revision_query	N/A
Error Query	*Prefix*_error_query	Utility
Error Message	*Prefix*_error_message	N/A
Get Specific Driver C Handle	*Prefix*_GetSpecificDriverCHandle	N/A
Get Specific Driver IUnknown Pointer	*Prefix*_GetSpecificDriverIUnknownPtr	N/A
Get Error	*Prefix*_GetError	N/A
Clear Error	*Prefix*_ClearError	N/A
Lock Session	*Prefix*_LockSession	Utility
Unlock Session	*Prefix*_UnlockSession	Utility
Get Next Coercion Record	*Prefix*_GetNextCoercionRecord	DriverOperation
Get Next Interchange Warning	*Prefix*_GetNextInterchangeWarning	DriverOperation
Reset Interchange Check	*Prefix*_ResetInterchangeCheck	DriverOperation
Clear Interchange Warnings	*Prefix*_ClearInterchangeWarnings	DriverOperation
Close	*Prefix*_close	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.

4.4 Relationship of Inherent Attributes and Different Types of IVI Driver

Some inherent attributes are exported by all types of IVI drivers— IVI.NET specific 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.NET specific drivers, 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.NET specific drivers and 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

.NET Property Name

DriverOperation.Cache

COM Property Name

DriverOperation.Cache

C Constant Name

*PREFIX*_ATTR_CACHE

Description

If True, the specific driver caches the value of attributes, and the IVI specific driver keeps track of the current instrument settings so that it can avoid sending redundant commands to the instrument. If False, the specific driver does not cache the value of attributes.

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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

.NET Property Name

N/A

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

.NET Property Name

N/A

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

.NET Property Name

N/A

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 characters including the NULL character.

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

.NET Property Name

N/A

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 characters including the NULL character.

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

.NET Property Name

N/A

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 characters including the NULL character.

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

.NET Property Name

N/A

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 characters including the NULL character.

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 (IVI-C & IVI-COM Only)

Data Type	Access
ViString	RO

.NET Property Name

N/A
(See the Get Group Capabilities method.)

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 & IVI.NET Only)

Data Type	Access
ViInt32	RO

.NET Property Name

Identity.SpecificationMajorVersion

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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

Data Type	Access
ViInt32	RO

.NET Property Name

Identity.SpecificationMinorVersion

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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.11 Component Description (IVI-COM & IVI.NET Only)

Data Type	Access
ViString	RO

.NET Property Name

Identity.Description

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.]
```

For IVI-COM, the string that this attribute returns contains a maximum of 256 characters including the NULL character. For IVI.NET, the string that this attribute returns has no maximum size.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.12 Component Identifier (IVI-COM & IVI.NET Only)

Data Type	Access
ViString	RO

.NET Property Name

Identity.Identifier

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 characters including the NULL character.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.13 Component Revision (IVI-COM & IVI.NET Only)

Data Type	Access
ViString	RO

.NET Property Name

Identity.Revision

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.

For IVI-COM, the string that this attribute returns contains a maximum of 256 characters including the NULL character. For IVI.NET, the string that this attribute returns has no maximum size.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.14 Component Vendor (IVI-COM & IVI.NET Only)

Data Type	Access
ViString	RO

.NET Property Name

Identity.Vendor

COM Property Name

Identity.Vendor

C Constant Name

N/A

Description

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

For IVI-COM, the string that this attribute returns contains a maximum of 256 characters including the NULL character. For IVI.NET, the string that this attribute returns has no maximum size.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.15 Driver Setup

Data Type	Access
ViString	RO

.NET Property Name

DriverOperation.DriverSetup

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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.16 I/O Resource Descriptor

Data Type	Access
ViString	RO

.NET Property Name

DriverOperation.IoResourceDescriptor

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 characters including the NULL character.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

.NET Property Name

N/A

(An IVI.NET specific driver is always initialized. See section 4.1, *.NET Inherent Capabilities*, and section 8, *IVI.NET Specific Driver Constructor* for details.)

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

.NET Property Name

Identity.InstrumentFirmwareRevision

COM Property Name

Identity.InstrumentFirmwareRevision

C Constant Name

*PREFIX*_ATTR_INSTRURMENT_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 or a string indicating that it cannot query the instrument identity.

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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.19 Instrument Manufacturer

Data Type	Access
ViString	RO

.NET Property Name

Identity.InstrumentManufacturer

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 or a string indicating that it cannot query the instrument identity.

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.

For IVI-C and IVI-COM, the string that this attribute returns contains a maximum of 256 characters including the NULL character. For IVI.NET, the string that this attribute returns has no maximum size.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.20 Instrument Model

Data Type	Access
ViString	RO

.NET Property Name

Identity.InstrumentModel

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 or a string indicating that it cannot query the instrument identity.

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.

For IVI-C and IVI-COM, the string that this attribute returns contains a maximum of 256 characters including the NULL character. For IVI.NET, the string that this attribute returns has no maximum size.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.21 Interchange Check

Data Type	Access
ViBoolean	R/W

.NET Property Name

NA

See the IVI.NET Interchange Check Warning Event.

COM Property Name

DriverOperation.InterchangeCheck

C Constant Name

*PREFIX*_ATTR_INTERCHANGE_CHECK

Description

If True, the specific driver performs interchangeability checking. For C and COM, 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. For .NET, if the Interchange Check attribute is enabled, the specific driver raises an Interchange Check Warning Event if any user has registered for the event. Refer to Section 9.1.2, *Interchange Check Warning Event (IVI.NET Only)*, for more information. If False, the specific driver does not perform interchangeability checking.

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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

.NET Property Name

DriverOperation.LogicalName

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 characters including the NULL character.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.23 Query Instrument Status

Data Type	Access
ViBoolean	R/W

.NET Property Name

DriverOperation.QueryInstrumentStatus

COM Property Name

DriverOperation.QueryInstrumentStatus

C Constant Name

*PREFIX*_ATTR_QUERY_INSTRUMENT_STATUS

Description

If True, the IVI specific driver queries the instrument status at the end of each user operation. If False, the IVI specific driver does not query 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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

.NET Property Name

DriverOperation.RangeCheck

COM Property Name

DriverOperation.RangeCheck

C Constant Name

*PREFIX*_ATTR_RANGE_CHECK

Description

If True, the IVI specific driver validates attribute values and function parameters. If False, the IVI specific driver does not validate attribute values and function parameters.

If range check is 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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

.NET Property Name

N/A

See the IVI.NET Coercion Record Event.

COM Property Name

DriverOperation.RecordCoercions

C Constant Name

*PREFIX*_ATTR_RECORD_COERCIONS

Description

If True, the IVI specific driver keeps a list of the value coercions it makes for ViInt32 and ViReal64 attributes. If False, the IVI specific driver does not keep 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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

.NET Property Name

DriverOperation.Simulate

COM Property Name

DriverOperation.Simulate

C Constant Name

*PREFIX*_ATTR_SIMULATE

Description

If True, the IVI specific driver simulates instrument driver I/O operations. If False, the IVI specific driver communicates directly with the instrument.

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.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

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

.NET Property Name

N/A

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

.NET Property Name

N/A

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

.NET Property Name

N/A

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 characters including the NULL character.

5.30 Specific Driver Locator (IVI-C Only)

Data Type	Access
ViString	RO

.NET Property Name

N/A

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

.NET Property Name

N/A

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 `fl45_init`, then `fl45` is the prefix for that driver.

The string that this attribute returns contains a maximum of 32 characters including the NULL character.

5.32 Specific Driver Revision (IVI-C Only)

Data Type	Access
`ViString`	RO

.NET Property Name

N/A

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 characters including the NULL character.

5.33 Specific Driver Vendor (IVI-C Only)

Data Type	Access
ViString	RO

.NET Property Name

N/A

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 characters including the NULL character.

5.34 Supported Instrument Models (IVI-C & IVI-COM Only)

Data Type	Access
ViString	RO

.NET Property Name

N/A
(See the Get Supported Instrument Models method.)

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.

.NET Method Prototype

N/A

COM Method Prototype

N/A

C Function Prototype

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

Parameters

Inputs	Description	Data Type
`Vi`	Unique identifier for an IVI session. The user can pass `VI_NULL`.	`ViSession`

Return Values (C)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

6.2 Clear Interchange Warnings (IVI-C & IVI-COM Only)

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.

.NET Method Prototype

```
N/A.
```

(See the Interchange Check Warning Event.)

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 (C/COM)

Section 11, *Common IVI-C and IVI-COM 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 in IVI-C and IVI-COM, 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.

When the user finishes using an IVI.NET driver, the user should call either the Close method or IDisposable.Dispose. The IVI.NET Close method shall call IDisposable.Dispose and take no other action. Note that this implies that the I/O connection is closed in IDisposable.Dispose rather than Close. In addition, all IVI.NET drivers shall implement Object.Finalize, which shall call Dispose. Refer to Microsoft documentation for IDisposable.Dispose for additional responsibilities and suggested implementation patterns of IDisposable.Dispose.

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.

.NET Method Prototype

```
void Close();
```

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 (C/COM)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

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.

.NET Method Prototype

```
void Utility.Disable();
```

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 (C/COM)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

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.

.NET Method Prototype

N/A

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
`Vi`	Unique identifier for an IVI session.	`ViSession`
`ErrorCode`	Instrument driver status code	`ViStatus`

Outputs	Description	Data Type
`ErrorMessage`	Instrument driver error message	`ViChar[]`

Return Values (C)

Section 11, *Common IVI-C and IVI-COM 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 NULL character, 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.

For IVI.NET, the method returns an object of type ErrorQueryResult, which is a struct that includes an Int32 *Code* property and a String *Message* property that correspond to the IVI-COM and IVI-C *ErrorCode* and *ErrorMessage* parameters, respectively.

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.

.NET Method Prototype

```
struct ErrorQueryResult
{
     Int32 Code {get}
     String Message {get}
}

ErrorQueryResult Utility.ErrorQuery();
```

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 (C/COM)`	Instrument error code	`ViInt32`
`ErrorMessage(C/COM)`	Instrument error message	`ViChar[]`
`Return Value (.NET)`	A struct that includes the instrument error code and error message.	`ErrorQueryResult`

Return Values (C/COM)

The table below defines specific status codes that this function returns. Section 11, *Common IVI-C and IVI-COM 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

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below defines specific exceptions for this method.

Exception	Description
Unexpected Response	Unexpected response from instrument.

The table below defines specific warning events for this method.

Warning	Description
Error Query Not Supported	The instrument does not support an error query operation.

Compliance Notes

1. IVI-C specific drivers shall not write more than 256 characters, including the NULL character, 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.

.NET Method Prototype

N/A

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 (C)

Section 11, *Common IVI-C and IVI-COM 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.

.NET Method Prototype

N/A

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 (C)

Section 11, *Common IVI-C and IVI-COM 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.

.NET Method Prototype

N/A

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 (C)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

6.10 Get Group Capabilities (IVI.NET Only)

Description

Returns a list of names of 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 list is returned as an array of strings.

If the IVI specific driver does not comply with an IVI class specification, the specific driver returns an array with zero elements.

.NET Method Prototype

```
String[] Identity.GetGroupCapabilities();
```

COM Method Prototype

N/A
(See the Class Group Capabilities attribute.)

C Function Prototype

N/A
(See the Class Group Capabilities attribute.)

Parameters

Outputs	Description	Data Type
`Return Value`	The list of class capability groups that the IVI specific driver implements.	`String[]`

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

6.11 *Get Next Coercion Record (IVI-C & IVI-COM Only)*

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.
```

.NET Method Prototype

N/A.
(See the Coercion Record Event.)

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 (C/COM)

Section 11, *Common IVI-C and IVI-COM 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.

6.12 Get Next Interchange Warning (IVI-C & IVI-COM Only)

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.

.NET Method Prototype

`N/A.`
(See the Interchange Check Warning Event.)

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 (C/COM)

Section 11, *Common IVI-C and IVI-COM 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.

6.13 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.

.NET Method Prototype

N/A

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 (C)

Section 11, *Common IVI-C and IVI-COM 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 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.

.NET Method Prototype

N/A

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 (C)

Section 11, *Common IVI-C and IVI-COM 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.

6.15 Get Supported Instrument Models (IVI.NET Only)

Description

Returns a list of names of instrument models with which the IVI specific driver is compatible. The list is returned as an array of strings. For example, this attribute might return the strings "TKTDS3012", "TKTDS3014", and "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 strings "TDS3012", "TDS3014", and "TDS3016".

.NET Method Prototype

```
String[] Identity.GetSupportedInstrumentModels();
```

COM Method Prototype

N/A
(See the Supported Instrument Models attribute.)

C Function Prototype

N/A
(See the Supported Instrument Models attribute.)

Parameters

Outputs	Description	Data Type
Return Value	The list of supported instrument models with which the IVI specific driver is compatible.	String[]

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

6.16 Initialize (IVI-C & IVI-COM Only)

Description

For IVI.NET, refer to section 8, *IVI.NET Specific Driver Constructor*, for details of driver initialization.

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 does not implement.
  - 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 VXI*plug&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 Options String parameter contains an assignment for the Driver Setup attribute, the Initialize function assumes that everything following "`DriverSetup=`" is part of the assignment. Therefore, the user is expected to place the Driver Setup assignment at the end of the Options String parameter. The value that the user passes in the Options String parameter for the Driver Setup attribute must contain only ASCII characters.

Each IVI specific driver defines it own meaning and valid values for the Driver Setup attribute. Many specific drivers ignore the value of the Driver Setup attribute. Other specific drivers use the Driver Setup 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 Driver Setup 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 Driver Setup 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.

.NET Method Prototype

N/A
(See section 8, *IVI.NET Specific Driver Constructor* for .NET initialization.)

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 OptionsString,
                                          ViSession *Vi);

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

Note: *Prefix*_init exists for compatibility with VXI*plug&play*. Calling *Prefix*_init is equivalent to calling *Prefix*_InitWithOptions with VI_NULL or an empty string for the OptionsString 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
OptionsString	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 (C/COM)

Section 11, *Common IVI-C and IVI-COM 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.

6.17 Invalidate All Attributes

Description

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

.NET Method Prototype

```
void DriverOperation.InvalidateAllAttributes();
```

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 (C/COM)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Compliance Notes

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

6.18 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.

For IVI.NET specific drivers, this function obtains a multithread lock for this instance of the driver. Before it does so, Lock Session waits until all other execution threads have released their locks or for the length of time specified by the maximum time parameter, whichever come first. The type of lock obtained depends upon the parameters passed to the specific driver constructor. See Section **Error! Reference source not found.** for details.

The user can use Lock Session with IVI-C or IVI.NET 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. For IVI-C drivers, the protection that this lock provides only applies to this instance of the driver. Multiple instances of the driver will not be protected from simultaneously accessing the instrument. For IVI.NET drivers, the scope of the lock is determined by the constructor parameters used to instantiate the driver.

It is important to note that this lock is ***not*** related to I/O locks such as the VISA resource locking mechanism.

With IVI-C and IVI.NET 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.

For IVI.NET, calls to Lock Session must always obtain the same lock that is used internally by the IVI.NET driver to guard individual method calls.

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.

.NET Method Prototype

```
IIviDriverLock Lock();
IIviDriverLock Lock(PrecisionTimeSpan maximumTime);
```

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
maximumTime (.NET)	Specifies the maximum length of time for the function to wait to acquire the lock before failing.	PrecisionTimeSpan

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

Defined Values for the MaximumTime Parameter (.NET)

Name	*Description*		
		Language	*Identifier*
Zero	The function returns immediately. If If the lock is not immediately available, the function throws an exception.		
		.NET	PrecisionTimeSpan.Zero
MaxValue	The function waits indefinitely to acquire the lock.		
		.NET	PrecisionTimeSpan.MaxValue

Return Values (C/COM)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

6.19 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.

.NET Method Prototype

```
void Utility.Reset();
```

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 (C/COM)

The table below defines specific status codes that this function returns. Section 11, *Common IVI-C and IVI-COM 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

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below defines specific exceptions for this method.

Exception	Description
Reset Not Supported	The instrument does not support the reset operation.

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.

6.20 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.

.NET Method Prototype

```
void DriverOperation.ResetInterchangeCheck();
```

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 (C/COM)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

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.

6.21 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.

.NET Method Prototype

```
void Utility.ResetWithDefaults ();
```

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 (C/COM)

The table below defines specific status codes that this function returns. Section 11, *Common IVI-C and IVI-COM 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

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below defines specific exceptions for this method.

Exception	Description
Reset Not Supported	The instrument does not support the reset operation.

6.22 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.

.NET Method Prototype

N/A
(See the Component Revision and Instrument Firmware Revision attributes.)

COM Method Prototype

N/A
(See the Component Revision and Instrument Firmware Revision attributes.)

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 (C)

The table below defines specific status codes that this function returns. Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

Name	COM Identifier	C Identifier
Revision Query Not	`S_IVI_NSUP_REV_QUERY`	`IVI_WARN_NSUP_REV_QUERY`

Supported		
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 NULL character, into the `DriverRev` output parameter.
2. IVI-C specific drivers shall not write more than 256 characters, including the NULL character, into the `InstrRev` output parameter.

6.23 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.

For IVI.NET, the method returns an object of type SelfTestResult, which is a struct that includes an Int32 *Code* property and a String *Message* property that correspond to the IVI-COM and IVI-C *TestResult* and *TestMessage* parameters, respectively.

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.

.NET Method Prototype

```
struct SelfTestResult
{
     Int32 Code { get }
     String Message { get }
}

SelfTestResult Utility.SelfTest();
```

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 (C/COM)`	Returns the numeric result from the self test operation (0 = no error, e.g. the test passed)	`ViInt16`
`TestMessage (C/COM)`	Returns the self test status message.	`ViChar[]`
`Return Value (.NET)`	A struct that includes the numeric result from the self test operation (0 = no error, e.g. the test passed) and self test status message.	`SelfTestResult`

Return Values (C/COM)

The table below defines specific status codes that this function returns. Section 11, *Common IVI-C and IVI-COM 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

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below defines specific exceptions for this method.

Exception	Description
Unexpected Response	Unexpected response from instrument.

The table below defines specific warning events for this method.

Warning	Description
Self Test Not Supported	The instrument does not support a self test operation.

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 NULL character, into the `TestMessage` output parameter.

6.24 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.

.NET Method Prototype

N/A

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 (C)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

6.25 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 and IVI.NET specific drivers, this function releases a lock that the Lock Session function acquires. Refer to Lock Session for additional information on IVI session locks.

.NET Method Prototype

void IIviDriverLock.Unlock();

COM Method Prototype

```
HRESULT Utlity.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 (C)`	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 (C/COM)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

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.

.NET Method Prototype

N/A

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 (C)

Section 11, *Common IVI-C and IVI-COM 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.

.NET Method Prototype

N/A

COM Method Prototype

N/A

C Function Prototype

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

ViStatus _VI_FUNC Prefix_AttachToExistingServiceProvider (
                                          size_t 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 (C)

Section 11, *Common IVI-C and IVI-COM 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 IVI-COM and IVI.NET Wrappers for IVI-C Specific Drivers

This section defines an additional interface that IVI-COM and IVI.NET 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 (IVI-COM) or GetServiceProvider (.NET) and contains a property named `NativeCHandle` and a method named `AttachToExistingCSession`.

The following table lists the COM GUID for the IIviClassWrapper interface.

Interface	GUID
`IIviClassWrapper`	{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
`AttachToExistingCSession`	Attach To Existing C Session	M
`NativeCHandle`	Native C Handle	P

7.2.1 Native C Handle (IVI-COM Only)

Data Type	Access
`ViSession`	`RO`

.NET Property Name

NativeCHandle

COM Property Name

`NativeCHandle`

C Constant Name

N/A

Description

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

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this property.

Compliance Notes

1. Only IVI-COM and IVI.NET 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 an IVI-COM or IVI.NET wrapper object to an existing IVI-C specific driver session.

.NET Method Prototype

```
void AttachToExistingCSession (Int32 Vi);
```

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 (COM)

Section 11, *Common IVI-C and IVI-COM Error and Completion Codes*, defines general status codes that this function can return.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Compliance Notes

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

8. IVI.NET Specific Driver Constructor

This section gives a complete description of the IVI.NET specific driver constructor.

Description

IVI.NET drivers do not have an Initialize method. Instead, IVI.NET drivers are initialized in the constructor for the main driver class of the driver.

The driver constructor 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 it is not possible to query the instrument for its identity, the driver ignores the parameter.
- 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 throws the Reset Not Supported exception.
- 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:
    1. Disables the class extension capability groups that the IVI specific driver does not implement.
    2. If the class specification(s) 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.
    3. 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 simulation is enabled in the configuration store or by the driverSetup parameter, the constructor 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 without taking any action.
- If the user passes True for the `reset` parameter and the instrument cannot perform a reset, the IVI specific driver throws the Reset Not Supported exception.
- 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.

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.NET driver constructor shall retrieve all the information for a session from the IVI configuration store. The IVI driver shall not read any information from the IVI configuration store for a session after the constructor 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 VXI*plug&play* specifications for the grammar of VISA resource descriptor strings.

The user can use the `options` 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 `options` 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 `options` 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 `options` 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 Option Name

Attribute	Default Initial Value	Option 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 `options` parameter is "*`Name=Value`*", where *`Name`* is one of the option names in the table above. Initialize interprets the *`Name`* and *`Value`* fields in a case-insensitive manner.

For the attributes of type `Boolean`, *`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 Options parameter contains an assignment for the Driver Setup attribute, the Initialize function assumes that everything following "`DriverSetup=`" is part of the assignment. Therefore, the user is expected to place the Driver Setup assignment at the end of the `options` parameter. The value that the user passes in the `options` parameter for the Driver Setup attribute must contain only ASCII characters.

Each IVI specific driver defines it own meaning and valid values for the Driver Setup attribute. Many specific drivers ignore the value of the Driver Setup attribute. Other specific drivers use the Driver Setup 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 Driver Setup attribute to allow the user to specify a particular instrument model to simulate.

Note that the constructor shall not raise warning events, including interchange check and coercion warning events, in the constructor.

The IVI specific driver ignores all white space in the `options` parameter outside the Driver Setup string.

.NET Constructor Prototypes

The IVI.NET specific driver shall implement two constructors with the following prototypes.

```
<DriverClassName> (String resourceName,
                   Boolean idQuery,
                   Boolean reset,
                   String options);


<DriverClassName> (String resourceName,
                   Boolean idQuery,
                   Boolean reset);
```

If the IVI.NET specific driver supports machine-wide multithread locking or process-wide locking across multiple driver instances, the specific driver shall also implement the following prototype.

```
<DriverClassName> (String resourceName,
                   Boolean idQuery,
                   Boolean reset,
                   LockType lockType,
                   String accessKey,
                   String options);
```

IVI.NET specific drivers may implement additional constructors. Any additional instrument specific constructors must include the resourceName parameter as the first parameter to the constructor.

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.	`String`
`idQuery`	Specifies whether to verify the ID of the instrument.	`Boolean`
`reset`	Specifies whether to reset the instrument.	`Boolean`
`lockType`	Specifies whether to use process-wide locking or machine-wide locking. Table 6.2 below explains how the value specified here is used in conjunction with the accessKey parameter to determine the kind of multithreaded lock to use for the driver instance. Refer to Section 4.3.11, *Multithread Safety*, of *IVI-3.1: Driver Architecture Specification* for a complete description of IVI.NET driver locking.	`Ivi.Driver.LockType`
`accessKey`	Specifies a user-selectable access key to identify the lock. Driver instances that are created with the same accessKey will be protected from simultaneous access by multiple threads within a process or across processes, depending upon the value of the lockType parameter. Table 6.2 below explains how the accessKey is used in conjunction with the lockType parameter to determine the kind of multithreaded lock to use for the driver instance. Refer to Section 4.3.11, *Multithread Safety*, of *IVI-3.1: Driver Architecture Specification* for a complete description of IVI.NET driver locking.	`String`

options	A string that allows the user to specify the initial values of certain inherent attributes.	String

It is possible that different client applications or different threads within an application may attempt to create instances of IVI.NET drivers with locking requirements that conflict with other instances of the driver. These conflicts occur if the driver is instantiated with the same value for the accessKey parameter but different values for the lockType parameter. Table 6-2 below explains how IVI.NET drivers are required to resolve these conflicts. In all cases, the resulting lock type must not change over the lifetime of the driver instance.

-

Table 6-2. Required Lock Type Behavior for Drivers With the Same Access Key

Requested Lock Type	Existing Lock Type	Resulting Lock Type
Process	Process	Process
Machine	Machine	Machine
Process	Machine	Machine
Machine	Process	Error Driver throws InvalidOperationException.

As the table shows, there are two cases in which a conflict occurs. If a driver requests a process-wide lock and a machine-wide lock has already been created by a different instance of the IVI.NET driver (with the same access key), then the IVI.NET driver shall "promote" the lock requested by the client and create a machine-wide lock instead of the requested process-wide lock. This ensures that the degree of locking established by existing driver instances is honored.

The second conflict occurs if a machine-wide lock is requested by the client but another instance of the same IVI.NET driver (with the same access key) has already been created with a process-wide lock. Since the type of lock used by an IVI.NET driver cannot change over the lifetime of the driver instance, there is no way to "promote" the lock type of the existing driver instance and, thus no way to comply with the new instance's requested degree of locking. Consequently, the constructor of the newly created instance must throw an InvalidOperationException.

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, by this method.

The table below defines specific exceptions for this constructor.

Exception	Description
Bad Option Name	An option name in the option string is unknown.
Bad Option Value	An value in the option string is invalid.
ID Query Failed	Instrument ID query failed.
Missing Option Name	The option string is missing an option name.
Missing Option Value	The option string is missing an option value.

Exception	Description
Reset Failed	Instrument reset failed.
Reset Not Supported	The instrument does not support the reset operation.
Resource Unknown	Unknown resource.

9. IVI.NET Event Descriptions

This section gives a complete description of each IVI.NET inherent event.

9.1 IVI.NET Events

IVI.NET defines the following events in support of the inherent capabilities

- Coercion Record Event (IVI.NET Only).
- Interchange Check Warning Event (IVI.NET Only).
- Driver Warning Event (IVI.NET Only).

9.1.1 Coercion Record Event (IVI.NET Only)

Description

This event is fired whenever the driver creates a coercion record. Clients who have registered as listeners will receive the event.

The `Text` property of the `CoercionEventArgs` shall contain the following information:

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

A recommended format for the `Text` property of the `CoercionEventArgs` is as follows:

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

And example `Text` property of the `CoercionEventArgs` is as follows:

```
Property VerticalRange on channel ch1 was coerced from 9.0 to 10.0.
```

.NET Event Prototype

```
class CoercionEventArgs : EventArgs
{
    String Text { get }
}
event EventHandler<CoercionEventArgs> Coercion;
```

C & COM Prototypes

`N/A`
(See the Record Value Coercions property/attribute and the Get Next Coercion Record method/function.)

Event Arguments

(The name of the event arguments type for this event is `CoercionEventArgs`.)

Member	Description	Data Type
`Text`	The text of a coercion record..	`String`

.NET Exceptions

Section **Error! Reference source not found.**, *Error! Reference source not found.*, defines general exceptions that may be thrown, and warning events that may be raised, when the event handler is registered.

If the driver does not support Coercion events, the driver throws an Event Not Supported exception when the client tries to register to receive Coercion events.

9.1.2 Interchange Check Warning Event (IVI.NET Only)

Description

This event is fired whenever the driver creates an interchange check warning. Clients who have registered as listeners will receive the event.

.NET Event Prototype

```
    class InterchangeCheckWarningEventArgs : EventArgs
    {
        String Text { get }
    }
event EventHandler<InterchangeCheckWarningEventArgs> InterchangeCheckWarning;
```

C & COM Prototypes

N/A

See the Interchange Check property/attribute, the Get Next Interchange Warning method/function, and the Clear Interchange Warnings method/function.

Event Arguments

The name of the event arguments type for this event is `InterchangeCheckWarningEventArgs`.

Member	Description	Data Type
`Text`	The text of an interchange check warning.	`String`

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown, and warning events that may be raised, when the event handler is registered.

If the driver does not support Interchange events, the driver throws an Event Not Supported exception when the client tries to register to receive Interchange events.

9.1.3 Driver Warning Event (IVI.NET Only)

Description

This event is fired whenever the driver creates a driver warning. Clients who have registered as listeners will receive the event.

.NET Event Prototype

```
    class WarningEventArgs : EventArgs
    {
        Guid Code { get }
        String Text { get }
    }

event EventHandler<WarningEventArgs> Warning;
```

C & COM Prototypes

N/A

In IVI-C and IVI-COM, warnings are returned as positive return values.

Event Arguments

The name of the event arguments type for this event is `WarningEventArgs`.

Member	Description	Data Type
`Code`	A GUID that uniquely identifies the warning.	`Guid`
`Text`	The text of a driver warning.	`String`

.NET Exceptions

Section **Error! Reference source not found.**, ***Error! Reference source not found.***, defines general exceptions that may be thrown when the event handler is registered.

If the driver does not support Interchange events, the driver throws an Event Not Supported exception when the client tries to register to receive Interchange events.

10. 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.

10.1 Inherent Attribute ID Values

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

Attribute Name	ID Value
*PREFIX*_ATTR_RANGE_CHECK	IVI_INHERENT_ATTR_BASE + 2
*PREFIX*_ATTR_QUERY_INSTRUMENT_STATUS	IVI_INHERENT_ATTR_BASE + 3
*PREFIX*_ATTR_CACHE	IVI_INHERENT_ATTR_BASE + 4
*PREFIX*_ATTR_SIMULATE	IVI_INHERENT_ATTR_BASE + 5
*PREFIX*_ATTR_RECORD_COERCIONS	IVI_INHERENT_ATTR_BASE + 6
*PREFIX*_ATTR_DRIVER_SETUP	IVI_INHERENT_ATTR_BASE + 7
*PREFIX*_ATTR_INTERCHANGE_CHECK	IVI_INHERENT_ATTR_BASE + 21
*PREFIX*_ATTR_CLASS_DRIVER_PREFIX	IVI_INHERENT_ATTR_BASE + 301
*PREFIX*_ATTR_SPECIFIC_DRIVER_PREFIX	IVI_INHERENT_ATTR_BASE + 302
*PREFIX*_ATTR_SPECIFIC_DRIVER_LOCATOR	IVI_INHERENT_ATTR_BASE + 303
*PREFIX*_ATTR_IO_RESOURCE_DESCRIPTOR	IVI_INHERENT_ATTR_BASE + 304
*PREFIX*_ATTR_LOGICAL_NAME	IVI_INHERENT_ATTR_BASE + 305
*PREFIX*_ATTR_SUPPORTED_INSTRUMENT_MODELS	IVI_INHERENT_ATTR_BASE + 327
*PREFIX*_ATTR_GROUP_CAPABILITIES	IVI_INHERENT_ATTR_BASE + 401
*PREFIX*_ATTR_INSTRUMENT_FIRMWARE_REVISION	IVI_INHERENT_ATTR_BASE + 510
*PREFIX*_ATTR_INSTRUMENT_MANUFACTURER	IVI_INHERENT_ATTR_BASE + 511
*PREFIX*_ATTR_INSTRUMENT_MODEL	IVI_INHERENT_ATTR_BASE + 512
*PREFIX*_ATTR_SPECIFIC_DRIVER_VENDOR	IVI_INHERENT_ATTR_BASE + 513
*PREFIX*_ATTR_SPECIFIC_DRIVER_DESCRIPTION	IVI_INHERENT_ATTR_BASE + 514
*PREFIX*_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION	IVI_INHERENT_ATTR_BASE + 515
*PREFIX*_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION	IVI_INHERENT_ATTR_BASE + 516
*PREFIX*_ATTR_CLASS_DRIVER_VENDOR	IVI_INHERENT_ATTR_BASE + 517
*PREFIX*_ATTR_CLASS_DRIVER_DESCRIPTION	IVI_INHERENT_ATTR_BASE + 518
*PREFIX*_ATTR_CLASS_DRIVER_CLASS_SPEC_MAJOR_VERSION	IVI_INHERENT_ATTR_BASE + 519
*PREFIX*_ATTR_CLASS_DRIVER_CLASS_SPEC_MINOR_VERSION	IVI_INHERENT_ATTR_BASE + 520
*PREFIX*_ATTR_SPECIFIC_DRIVER_REVISION	IVI_INHERENT_ATTR_BASE + 551

Attribute Name	ID Value
*PREFIX*_ATTR_CLASS_DRIVER_REVISION	IVI_INHERENT_ATTR_BASE + 552

10.2 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

10.3 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
IVI_INHERENT_ATTR_BASE + 321
IVI_INHERENT_ATTR_BASE + 601
IVI_INHERENT_ATTR_BASE + 602
IVI_INHERENT_ATTR_BASE + 603
IVI_INHERENT_ATTR_BASE + 704
IVI_INHERENT_ATTR_BASE + 708
IVI_INHERENT_ATTR_BASE + 801

10.4 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
IVI_INHERENT_ATTR_BASE + 203

11. Common IVI-C and IVI-COM 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*.

11.1 IVI-C and IVI-COM Error and Completion Codes

The following table lists error and completion codes returned by IVI-C and IVI-COM 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

11.2 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."** ***%s1 = Attribute ID***
Attribute Not Writeable	IVI_ERROR_ATTR_NOT_WRITEABLE	**"%s: Attribute %s1 is read only."** ***%s1 = Attribute name***
Attribute Not Readable	IVI_ERROR_ATTR_NOT_READABLE	**"%s: Attribute %s1 is write only."** ***%s1 = Attribute name***
Invalid Value	IVI_ERROR_INVALID_VALUE	**"%s: Invalid value (%s1) for function %s2, parameter %s3."** ***%s1 = out-of-range value*** ***%s2 = function name*** ***%s3 = parameter name***
Function Not Supported	IVI_ERROR_FUNCTION_NOT_SUPPORTED	**"%s: Does not support this class-compliant feature: function %s1."** ***%s1 = function name***
Attribute Not Supported	IVI_ERROR_ATTRIBUTE_NOT_SUPPORTED	**"%s: Does not support this class-compliant feature: attribute %s1."** ***%s1 = attribute name***

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

Name	C Identifier	C Message String
Value Not Supported	IVI_ERROR_VALUE_NOT_SUPPORTED	**"%s: Does not support this class-compliant feature: (enumeration) value %s1 passed as the value for parameter %s2 in function %s3."** ***%s1 = enumeration value name* or *value*** ***%s2 = parameter name*** ***%s3 = function name*** **"%s: Does not support this class-compliant feature: (enumeration) value %s1 passed as the value for attribute %s2."** ***%s1 = enumeration value name* or *value*** ***%s2 = attribute name***
Types Do Not Match	IVI_ERROR_TYPES_DO_NOT_MATCH	**"%s: SetAttribute%s1 called for attribute of type %s2."** **"%s: GetAttribute%s1 called for attribute of type %s2."** ***%s1 =data type of attribute access function*** ***%s2 =data type of attribute***
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	**"%s: The %s1 name in the option string is unknown."** ***%s1 = bad option name***
Bad Option Value	IVI_ERROR_BAD_OPTION_VALUE	**"%s: The %s1 value in the option string is unknown."** ***%s1 = bad option value***
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."**

11.3 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."** ***%s1 = out-of-range value*** ***%s2 = method name*** ***%s3 = parameter name***
Function Not Supported	E_IVI_METHOD_NOT_SUPPORTED	**"%s: Does not support this class-compliant feature: method %s1."** ***%s1 = method name***
Attribute Not Supported	E_IVI_PROPERTY_NOT_SUPPORTED	**"%s: Does not support this class-compliant feature: property %s1."** ***%s1 = property name***
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."** ***%s1 = enumeration value name* or *value*** ***%s2 = parameter name*** ***%s3 = method name*** **"%s: Does not support this class-compliant feature: (enumeration) value %s1 passed as the value for property %s2."** ***%s1 = enumeration value name* or *value*** ***%s2 = property name***
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."** ***%s1 = bad option name***
Bad Option Value	E_IVI_BAD_OPTION_VALUE	**"%s: The %s1 value in the option string is unknown."** ***%s1 = bad option value***
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."** **%s1 = method name** ***%s2 = parameter name***
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."** **%s1 = repeated capability name**
Invalid Range in Selector	E_IVI_INVALID_RANGE_IN_SELECTOR	**"%s: The range %s1 is not valid for the repeated capability %s2."** **%s1 = range** **%s2 = repeated capability name**

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 selectory"**
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."**

11.4 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

11.5 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.

11.6 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_CANNNOT_RECOVER

12. Common IVI.NET Exceptions and Warnings

This section defines the list of IVI.NET exceptions and warnings. For general information on IVI.NET exceptions and warnings, refer to IVI-3.1: *Driver Architecture Specification*.

12.1 General Information About Exceptions

All IVI defined exceptions derive from System.Exception. The public constructors from System.Exception are preserved, including parameter names and semantics.

Constructors with additional, exception specific parameters are added when appropriate. All parameters that are not inherited from System.Exception are documented with the exception. In general, constructors that provide additional parameters are recommended for use in IVI.NET native drivers. Including parameter information with exceptions provides users with additional useful information about exceptions that is consistent across drivers.

IVI defined exceptions implement read-only properties for all constructor parameters that are added for the exception.

Constructors that take a string 'message' parameter are recommended for use when the default message is not sufficient, or when the code calling the excpetion can provide better information. For example, when creating an IVI.NET driver that wraps an IVI-C driver, the .NET wrapper may not have access to additional parameters, and use the error message created by the IVI-C driver instead.

Depending on the constructor used by to create the exception, the value of the additional parameters may or may not have been set. Clients should not write code that assumes that additional parameters have been set.

The exception's Message property concatenates (1) either a default message or the value from the constructor's 'message' parameter, and (2) the name and value of each additional parameter that has been set, one parameter per line. The default message string documented in this specification includes a line for each parameter that could possibly be set, but if a parameter is not set, that parameter name and value are not included in the message.

Because the exception message content may vary, clients should not assume a standard format for message strings. For common uses of exceptions, users should be able to tell what they need to know from the exception type, so that parsing the message string is unnecessary.

IVI.NET drivers may throw exceptions that are derived from inherent or class-compliant exceptions from inherent, class-compliant or instrument specific interfaces.

12.2 Mapping IVI-C and IVI-COM Error Codes to IVI.NET

IVI.NET exceptions are designed to reflect IVI.NET paradigms. For this reason, there is not a one-to-one mapping from IVI-C and IVI-COM exceptions to IVI.NET exceptions. The following table describes the mapping and the reasons for significant differences.

IVI-C/IVI-COM Name	IVI.NET Exception
Success	N/A – Success exceptions do not exist.
Cannot Recover	N/A – IVI.NET drivers that wrap IVI-C or IVI-COM drivers should use `Ivi.Driver.IviCDriverException` or `Ivi.Driver.IviComDriverException`.
Instrument Status	`Ivi.Driver.InstrumentStatusException`
Cannot Open File	N/A - Let the framework exceptions thrown when opening a file filter up to the user.
Error Reading File	N/A - Let the framework exceptions thrown when reading a file filter up to the user.
Error Writing File	N/A - Let the framework exceptions thrown when writing a file filter up to the user.
Invalid Path Name	N/A - Let the framework exceptions thrown when using a file path filter up to the user.
Invalid Value	`Ivi.Driver.OutOfRangeException`
Function Not Supported	`Ivi.Driver.OperationNotSupportedException`
Attribute Not Supported	`Ivi.Driver.OperationNotSupportedException`
Value Not Supported	`Ivi.Driver.ValueNotSupportedException`
Not Initialized	N/A – IVI.NET drivers are always initialized in the constructor.
Unknown Channel Name	`Ivi.Driver.SelectorNameException`
Too Many Open Files	N/A - Let the framework exceptions thrown when opening a file filter up to the user.
Channel Name Required	`Ivi.Driver.SelectorNameRequiredException`
Missing Option Name	`Ivi.Driver.OptionMissingException`
Missing Option Value	`Ivi.Driver.InvalidOptionValueException`
Bad Option Name	`Ivi.Driver.UnknownOptionException`
Bad Option Value	`Ivi.Driver.InvalidOptionValueException`
Out of Memory	`System.InsufficientMemoryException`
Operation Pending	`Ivi.Driver.OperationPendingException`
Null Pointer	`System.ArgumentNullException`
Unexpected Response	`Ivi.Driver.UnexpectedResponseException`
File Not Found	`System.IO.FileNotFoundException`
Invalid File Format	`Ivi.Driver.FileFormatException`
Status Not Available	N/A - Use either `Ivi.Driver.OperationPendingException` or `Ivi.Driver.OperationNotSupportedException`
ID Query Failed	`Ivi.Driver.IdQueryFailedException`
Reset Failed	`Ivi.Driver.ResetFailedException`
Resource Unknown	N/A – If the VISA Open (or similar I/O call) does not succeed, throw an `Ivi.Driver.IOException`.
Already Initialized	N/A – IVI.NET drivers are always initialized in the constructor.

IVI-C/IVI-COM Name	IVI.NET Exception
Cannot Change Simulation State	`Ivi.Driver.SimulationStateException`
Invalid Number of Levels in Selector	`Ivi.Driver.SelectorHierarchyException`
Invalid Range in Selector	`Ivi.Driver.SelectorRangeException`
Unknown Name in Selector	`Ivi.Driver.SelectorNameException`
Badly-Formed Selector	`Ivi.Driver.SelectorFormatException`
Unknown Physical Identifier	`Ivi.Driver.UnknownPhysicalNameException`
ID Query Not Supported	N/A (warning)
Reset Not Supported	`Ivi.Driver.ResetNotSupportedException`
Self Test Not Supported	N/A (warning)
Error Query Not Supported	N/A (warning)
Revision Query Not Supported	N/A (warning)
N/A	`Ivi.Driver.ConfigurationServerException`
N/A	`Ivi.Driver.IOException`
N/A	`Ivi.Driver.IOTimeoutException`
N/A	`Ivi.Driver.IviCDriverException`
N/A	`Ivi.Driver.OptionStringFormatException`
Max Time Exceeded (IVI-3.3)	`Ivi.Driver.MaxTimeExceededException`
TriggerNotSoftware (IVI-3.3)	`Ivi.Driver.TriggerNotSoftwareException`

12.3 Mapping IVI-COM Session Factory Error Codes to IVI.NET

IVI.NET exceptions are designed to reflect IVI.NET paradigms. For this reason, there is not a one-to-one mapping from IVI-COM Session Factory exceptions to IVI.NET exceptions. The following table describes the mapping, bearing in mind that some mapped items (for instance, Driver Session Not Registered and the InvalidClassNameException) really have different meanings in COM and .NET, although the roles they play are analogous.

Refer to Section 3, *Error and Completion Code Value Definitions*, of *IVI-3.6: COM Session Factory Specification*, for a list of IVI-COM session factory error codes.

IVI-COM Name	IVI.NET Exception
No Prog ID	`Ivi.Driver.ClassNameNotFoundException`
No Config Store	`Ivi.Driver.ConfigurationStoreLoadException`
Driver Session Not Registered	`Ivi.Driver.InvalidClassNameException`
No Driver Session	`Ivi.Driver.SessionNotFoundException`

IVI-COM Name	IVI.NET Exception
No Software Module	`Ivi.Driver.SoftwareModuleNotFoundException`
N/A	`Ivi.Driver.DriverClassCreationException`

12.4 Common Exceptions

The following .NET Framework exceptions may be explicitly thrown by IVI.NET drivers where applicable.

- System.Argument NullException
- System.FileNotFoundException
- System.InsufficientMemoryException

Other exceptions defined by the .NET Framework may be explicitly thrown by IVI.NET drivers if IVI specifications do not specify a suitable exception. Other exceptions defined by the .NET Framework may be thrown by IVI.NET drivers by the .NET runtime or other libraries used by the drivers.

Common IVI.NET exceptions are defined in this specification and declared in the Ivi.Driver namespace.

- ConfigurationServerException
- FileFormatException
- IdQueryFailedException
- InstrumentStatusException
- InvalidOptionValueException
- IOException
- IOTimeoutException
- IviCDriverException
- IviComDriverException (Reserved)
- MaxTimeExceededException
- OperationNotSupportedException
- OperationPendingException
- OptionMissingException
- OptionStringFormatException
- OutOfRangeException
- ResetFailedException
- ResetNotSupportedException
- SelectorFormatException
- SelectorHierarchyException
- SelectorNameException
- SelectorNameRequiredException
- SelectorRangeRequiredException
- SimulationStateException
- TriggerNotSoftwareException
- UnexpectedResponseException
- UnknownOptionException
- UnknownPhysicalNameException
- ValueNotSupportedException

12.4.1 System.ArgumentNullException (.NET Framework)

Description

See the MSDN documentation for `System.ArgumentNullException`.

Exception

`System.ArgumentNullException`

Default Message String

```
Value cannot be null.
Parameter name: <paramName>.
```

Parameters

See the MSDN documentation for `System.ArgumentNullException`.

12.4.2 System.InsufficientMemoryException

Description

See the MSDN documentation for `System.InsufficientMemoryException`.

Exception

`System.InsufficientMemoryException`

Default Message String

`Insufficient memory to continue the execution of the program.`

Parameters

See the MSDN documentation for `System.InsufficientMemoryException`.

Usage

Note that `System.OutOfMemoryException` should only be thrown by the .NET runtime – it should never be thrown by IVI.NET drivers.

12.4.3 System.IO.FileNotFoundException

Description

See the MSDN documentation for `System.IO.FileNotFoundException`.

Exception

`System.IO.FileNotFoundException`

Default Message String

```
Unable to find the specified file.
File name: <fileName>.
```

Parameters

See the MSDN documentation for `System.IO.FileNotFoundException`.

12.4.4 ConfigurationServerException

Description

An error occurred while using the Configuration Server.

When accessing the IVI-COM Configuration Server using the primary interop assembly (PIA), this exception is used to relay an exception thrown by the configuration server PIA (for example, an Unauthorized Access exception or an IO exception). The exception thrown by the Configuration Server is the inner exception for this one.

When accessing the IVI-COM or IVI-C Configuration Server using other forms of interop, this exception is used to relay the error return code reported by the Configuration Server.

Exception

```
Ivi.Driver.ConfigurationServerException
```

Constructors

```
Ivi.Driver.ConfigurationServerException(System.Exception innerException);

Ivi.Driver.ConfigurationServerException(Int32 errorCode);

Ivi.Driver.ConfigurationServerException();

Ivi.Driver.ConfigurationServerException(String message);

Ivi.Driver.ConfigurationServerException(String message,
                                        System.Exception innerException);
```

Default Message String

```
An error occurred while using the Configuration Server.
Error code: <errorCode>
```

Parameters

Inputs	Description	Base Type
`errorCode`	The error code returned from the Configuration Server property or method when a .NET Configuration Server or PIA is not used.	`Int32`
`innerException`	The exception thrown by the Configuration Server (or Configuration Server PIA) that is the cause of the current exception. If the innerException parameter is not null, the current exception is raised in a catch block that handles the inner exception.	`System.Exception`

Usage

Since the driver is required to read all relevant configuration store information in the constructor, this exception shall only be thrown by the constructor.

If driver developers specify the message string, they are responsible for message string localization.

12.4.5 FileFormatException

Description

A file does not conform to it's expected format.

Exception

```
Ivi.Driver.FileFormatException
```

Constructors

```
Ivi.Driver.FileFormatException(Uri sourceUri);

Ivi.Driver.FileFormatException(Uri sourceUri,
                               System.Exception innerException);

Ivi.Driver.FileFormatException();

Ivi.Driver.FileFormatException(String message);

Ivi.Driver.FileFormatException(String message,
                               System.Exception innerException);
```

Default Message String

```
The file does not conform to the expected file format.
File URI: <sourceURI>
```

Parameters

Inputs	Description	Base Type
`sourceUri`	The URI of the file which is not formatted correctly.	`System.Uri`

Usage

If the driver catches an exception that prompted this exception (for example, a system File Not Found exception), that exception should be made the inner exception for this one.

If driver developers specify the message string, they are responsible for message string localization.

12.4.6 IdQueryFailedException

Description

The instrument ID query failed.

Exception

```
Ivi.Driver.IdQueryFailedException
```

Constructors

```
Ivi.Driver.IdQueryFailedException();

Ivi.Driver.IdQueryFailedException(String message);

Ivi.Driver.IdQueryFailedException(String message,
                                  System.Exception innerException);
```

Default Message String

```
The instrument ID query failed.
```

Usage

Under normal circumstances, an ID query is done once, either up-front in the constructor, or in the first get for a property that returns ID Query information. Class compliant properties that potentially return ID query information are InstrumentManufacturer, InstrumentModel, and InstrumentFirmwareRevision, which are all in the IiviDriverIdentity interface. Instrument specific properties, such as a property that returns serial number, may also do an ID query.

If driver developers specify the message string, they are responsible for message string localization.

12.4.7 InstrumentStatusException

Description

The driver detected an instrument error.

Exception

```
Ivi.Driver.InstrumentStatusException
```

Constructors

```
Ivi.Driver.InstrumentStatusException();

Ivi.Driver.InstrumentStatusException(String message);

Ivi.Driver.InstrumentStatusException(String message,
                                     System.Exception innerException);
```

Default Message String

```
Instrument error detected.  Use ErrorQuery() to determine the error(s).
```

Usage

Avoid using this exception to relay another exception. As a general rule, just let the original exception propagate up.

If driver developers specify the message string, they are responsible for message string localization.

12.4.8 InvalidOptionValueException

Description

An invalid value is assigned to an option.

Exception

```
Ivi.Driver.InvalidOptionValueException
```

Constructors

```
Ivi.Driver.InvalidOptionValueException(String optionName,
                                       String optionValue);

Ivi.Driver.InvalidOptionValueException();

Ivi.Driver.InvalidOptionValueException(String message);

Ivi.Driver.InvalidOptionValueException(String message,
                                       System.Exception innerException);
```

Default Message String

```
The option string contains an invalid option value.
Option name: <optionName>.
Option value: <optionValue>.
```

Parameters

Inputs	Description	Base Type
`optionName`	The name of the option.	`String`
`optionValue`	The invalid value assigned to the option.	`String`

Usage

Since the driver is required to process option strings in the constructor, this exception shall only be thrown from the constructor.

If driver developers specify the message string, they are responsible for message string localization.

12.4.9 IOException

Description

A call to the underlying I/O mechanism being used by the driver to communicate with the instrument has failed.

When accessing .NET I/O libraries or COM I/O libraries using a primary interop assembly (PIA), this exception is used to relay an exception thrown by the I/O library. The exception thrown by the I/O library is the inner exception for this one.

When accessing a native C or COM I/O library using other forms of interop, this exception is used to relay the error return code reported by the Configuration Server.

If the underlying I/O library reports a timeout, use the IOTimeoutException.

Exception

```
Ivi.Driver.IOException
```

Constructors

```
Ivi.Driver.IOException(System.Exception innerException);

Ivi.Driver.IOException(Int32 errorCode);

Ivi.Driver.IOException();

Ivi.Driver.IOException(String message);

Ivi.Driver.IOException(String message,
                       System.Exception Innerxception);
```

Default Message String

```
An instrument I/O error occurred.
```

Parameters

Inputs	Description	Base Type
`errorCode`	The error code returned from the I/O library property or method when a .NET I/O library or PIA is not used.	`Int32`
`innerException`	The exception thrown by the I/O library (or I/O library PIA) that is the cause of the current exception. If the innerException parameter is not null, the current exception is raised in a catch block that handles the inner exception.	`System.Exception`

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.10 IOTimeoutException

Description

A call to the underlying IO mechanism being used by the driver to communicate with the instrument has timed out.

Exception

```
Ivi.Driver.IOTimeoutException
```

Constructors

```
Ivi.Driver.IOTimeoutException(String message,
                              String timeout);

Ivi.Driver.IOTimeoutException();

Ivi.Driver.IOTimeoutException(String message);

Ivi.Driver.IOTimeoutException(String message,
                              System.Exception innerException);
```

Default Message String

```
An I/O timeout occurred.
Timeout: 2000mS.
```

Parameters

Inputs	Description	Base Type
`timeout`	The timeout that was exceeded, including units. For example, '2000 ms'	`String`

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.11 IviCDriverException

Description

When an underlying IVI-C driver was called to perform an action, the IVI-C driver action did not succeed.

Exception

```
Ivi.Driver.IviCInteropException
```

Constructors

```
Ivi.Driver.IviCInteropException(String message
                                Int32 errorCode);

Ivi.Driver.IviCDriverException();

Ivi.Driver.IviCInteropException(String message);

Ivi.Driver.IviCInteropException(String message,
                                System.Exception innerException);
```

Default Message String

```
The call to the IVI-C driver did not succeed.
```

Parameters

Inputs	Description	Base Type
`errorCode`	The status code returned by the IVI-C driver call.	`String`

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.12 IviComDriverException

Description

Reserved for future use.

12.4.13 MaxTimeExceededException

Description

The operation implemented by the method did not complete within the maximum time allowed.

Use this exception, rather than the IOTimeoutException, whenever a method includes a parameter (for example, maximumTime) that specifies maximum time allowed for the method's operation to complete.

Exception

```
Ivi.Driver.MaxTimeExceededException
```

Constructors

```
Ivi.Driver.MaxTimeExceededException(String message,
                                    String timeout);

Ivi.Driver.MaxTimeExceededException();

Ivi.Driver.MaxTimeExceededException(String message);

Ivi.Driver.MaxTimeExceededException(String message,
                                    System.Exception innerException);
```

Default Message String

```
The operation did not complete within the maximum time allowed.
Timeout: 2000mS.
```

Parameters

Inputs	Description	Base Type
`timeout`	The timeout that was exceeded, including units. For example, '2000 ms'	`String`

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.14 OperationNotSupportedException

Description

A driver feature (for this exception, a method, property, or event) is not supported by the driver.

Exception

```
Ivi.Driver.OperationNotSupportedException
```

Constructors

```
Ivi.Driver.OperationNotSupportedException(String message,
                                          String methodOrPropertyName);

Ivi.Driver.OperationNotSupportedException();

Ivi.Driver.OperationNotSupportedException(String message);

Ivi.Driver.OperationNotSupportedException(String message,
                                          System.Exception innerException);
```

Default Message String

```
The method or property is not supported.
Method or property name: <methodOrPropertyName>.
```

Parameters

Inputs	Description	Base Type
`methodOrPropertyName`	The name of the unsupported feature.	`String`

Usage

This exception should not be used for parameters. Use ValueNotSupportedException for enumeration values or discrete values from a list of defined values that aren't supported by the driver, and OutOfRangeException for other invalid values.

This exception should not be used for the Reset method. Use ResetNotSupportedException if the instrument does not support resets.

If driver developers specify the message string, they are responsible for message string localization.

12.4.15 OperationPendingException

Description

An operation is in progress that prevents the method or property from being executed.

Exception

```
Ivi.Driver.OperationPendingException
```

Constructors

```
Ivi.Driver.OperationPendingException();

Ivi.Driver.OperationPendingException(String message);

Ivi.Driver.OperationPendingException(String message,
                                     System.Exception innerException);
```

Default Message String

```
Operation in progress.
```

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.16 OptionMissingException

Description

A required option is missing from the option string.

Exception

```
Ivi.Driver.OptionMissingException
```

Constructors

```
Ivi.Driver.OptionMissingException(String message,
                                  String optionName);

Ivi.Driver.OptionMissingException();

Ivi.Driver.OptionMissingException(String message);

Ivi.Driver.OptionMissingException(String message,
                                  System.Exception innerException);
```

Default Message String

```
The option string is missing a required option.
Option name: <optionName>
```

Parameters

Inputs	Description	Base Type
`optionName`	The name of the missing option.	`String`

Usage

Since the driver is required to process option strings in the constructor, this exception shall only be thrown from the constructor.

If driver developers specify the message string, they are responsible for message string localization.

12.4.17 OptionStringFormatException

Description

The driver cannot parse the option string.

Exception

```
Ivi.Driver.OptionStringFormatException
```

Constructors

```
Ivi.Driver.OptionStringFormatException();

Ivi.Driver.OptionStringFormatException(String message);

Ivi.Driver.OptionStringFormatException(String message,
                                       System.Exception innerException);
```

Default Message String

```
The option string is not formatted correctly.
```

Usage

Since the driver is required to process option strings in the constructor, this exception shall only be thrown from the constructor.

If driver developers specify the message string, they are responsible for message string localization.

12.4.18 OutOfRangeException

Description

The driver detected an argument whose value is out or range.

Exception

```
Ivi.Driver.OutOfRangeException
```

Constructors

```
Ivi.Driver.OutOfRangeException(String paramName,
                               String actualValue
                               String range);

Ivi.Driver.OutOfRangeException();

Ivi.Driver.OutOfRangeException(String message);

Ivi.Driver.OutOfRangeException(String message,
                               System.Exception innerException);
```

Default Message String

```
The specified argument was out of the range of valid values.
Parameter name: <paramName>.
Actual value: <actualValue>.
Allowable range: <range>.
```

Parameters

Inputs	Description	Base Type
`paramName`	The name of the parameter to which the out of range value is assigned.	`String`
`actualValue`	The out of range value.	`String`
`range`	The allowable range	`String`

Usage

For a property set, the parameter name is “value”.

Use this exception only if a more specific exception is not appropriate.

If driver developers specify the message string, they are responsible for message string localization.

12.4.19 ResetFailedException

Description

The instrument reset failed.

Exception

```
Ivi.Driver.ResetFailedException
```

Constructors

```
Ivi.Driver.ResetFailedException();

Ivi.Driver.ResetFailedException(String message);

Ivi.Driver.ResetFailedException(String message,
                                System.Exception innerException);
```

Default Message String

```
The instrument reset failed.
```

Usage

Under normal circumstances, an instrument reset is done in the constructor and in the Reset() and ResetWithDefaults() methods in the IiviDriverUtility interface. For particular drivers, other properties and methods may include a reset if needed for some instrument specific reason.

If driver developers specify the message string, they are responsible for message string localization.

12.4.20 ResetNotSupportedException

Description

The instrument does not support the reset operation.

Exception

```
Ivi.Driver.ResetNotSupportedException
```

Constructors

```
Ivi.Driver.ResetNotSupportedException();

Ivi.Driver.ResetNotSupportedException(String message);

Ivi.Driver.ResetNotSupportedException(String message,
                                      System.Exception innerException);
```

Default Message String

```
The instrument does not support the reset operation.
```

Usage

If the instrument is capable of doing a reset, but the reset fails, use the Reset Failed exception.

If driver developers specify the message string, they are responsible for message string localization.

12.4.21 SelectorFormatException

Description

The selector is not a simple repeated capability selector, and the selector cannot be parsed.

If the selector is a complex selector that can be parsed, use one of the following exceptions

- SelectorNameException
- SelectorNameUnknownException
- SelectorRangeException
- SelectorHierarchyException

Exception

```
Ivi.Driver.SelectorFormatException
```

Constructors

```
Ivi.Driver.SelectorFormatException(String repeatedCapabilityName,
                                   String selectorValue);

Ivi.Driver.SelectorFormatException();

Ivi.Driver.SelectorFormatException(String message);

Ivi.Driver.SelectorFormatException(String message,
                                   System.Exception innerException);
```

Default Message String

```
Invalid format for repeated capability selector.
Repeated capability: <repeatedCapabilityName>
Repeated capability selector value: <selectorValue>
```

Parameters

Inputs	Description	Base Type
`repeatedCapabilityName`	The name of the repeated capability (not the repeated capability instance) associated with the method or property from which the exception was thrown.	`String`
`selectorValue`	The invalid selector value.	`String`

Usage

Since complex repeated capability selectors may not be used as indexers in IVI.NET, this exception should never be thrown by an indexer.

If the repeated capability selector does not support complex selectors, use SelectorNameException.

If driver developers specify the message string, they are responsible for message string localization.

12.4.22 SelectorHierarchyException

Description

A hierarchical repeated capability selector includes an invalid number of levels in the hierarchy of nested identifiers.

If the only problem is an unknown name or names in the range, the Selector Name Exception should be used.

Exception

```
Ivi.Driver.SelectorHierarchyException
```

Constructors

```
Ivi.Driver.SelectorHierarchyException(String repeatedCapabilityName,
                                      String selectorValue);

Ivi.Driver.SelectorHierarchyException();

Ivi.Driver.SelectorHierarchyException(String message);

Ivi.Driver.SelectorHierarchyException(String message,
                                      System.Exception innerException);
```

Default Message String

```
The repeated capability selector has the wrong number of levels.
Repeated capability: <repeatedCapabilityName>.
Repeated capability selector value: <selectorValue>.
```

Parameters

Inputs	Description	Base Type
`repeatedCapabilityName`	The name of the repeated capability (not the repeated capability instance) associated with the method or property from which the exception was thrown.	`String`
`selectorValue`	The repeated capability selector value that contains the invalid hierarchy.	`String`

Usage

Since complex repeated capability selectors may not be used as indexers in IVI.NET, this exception should never be thrown by an indexer.

If the repeated capability selector does not support complex selectors, use the Invalid Selector exception.

If driver developers specify the message string, they are responsible for message string localization.

12.4.23 SelectorNameException

Description

A repeated capability selector is expected, but the driver does not recognise the provided name.

This exception should be used with any repeated capability parameter or indexer when a more specific exception is not appropriate. More specific exceptions are:

- SelectorFormatException
- SelectorRangeException
- SelectorHierarchyException

Exception

```
Ivi.Driver.SelectorNameException
```

Constructors

```
Ivi.Driver.SelectorNameException(String repeatedCapabilityName,
                                 String selectorValue);

Ivi.Driver.SelectorNameException();

Ivi.Driver.SelectorNameException(String message);

Ivi.Driver.SelectorNameException(String message,
                                 System.Exception innerException);
```

Default Message String

```
Invalid repeated capability name in selector.
Repeated capability: <repeatedCapabilityName>.
Repeated capability selector value: <selectorValue>.
```

Parameters

Inputs	Description	Base Type
`repeatedCapabilityName`	The name of the repeated capability (not the repeated capability instance).	`String`
`selectorValue`	The repeated capability selector value that contains the invalid name.	`String`

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.24 SelectorNameRequiredException

Description

The selector has more than one instance of a repeated capability, but an instance is not specified. An empty string is only valid for a repeated capability selector if there is only one instance of the repeated capability.

Exception

```
Ivi.Driver.SelectorNameRequiredException
```

Constructors

```
Ivi.Driver.SelectorNameRequiredException(String message,
                                         String repeatedCapabilityName);

Ivi.Driver.SelectorNameRequiredException();

Ivi.Driver.SelectorNameRequiredException(String message);

Ivi.Driver.SelectorNameRequiredException(String message,
                                         System.Exception innerException);
```

Default Message String

```
The repeated capability selector name is required.
Repeated capability: <repeatedCapabilityName>.
```

Parameters

Inputs	Description	Base Type
`repeatedCapabilityName`	The name of the repeated capability (not the repeated capability instance).	`String`

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.25 SelectorRangeException

Description

A complex repeated capability selector includes an invalid range or list of repeated capability identifiers. This includes descending, repeated, overlapped ranges and lists.

If there is an invalid range or list of repeated capability identifiers in a hierarchical selector that also has an invalid number of levels, throw the SelectorHierarchyException.

If the only problem is an unknown name or names in the range, the SelectorNameException should be used.

Exception

```
Ivi.Driver.SelectorRangeException
```

Constructors

```
Ivi.Driver.SelectorRangeException(String repeatedCapabilityName,
                                  String selectorValue);

Ivi.Driver.SelectorRangeException();

Ivi.Driver.SelectorRangeException(String message);

Ivi.Driver.SelectorRangeException(String message,
                                  System.Exception innerException);
```

Default Message String

```
The repeated capability selector includes an invalid range or list.
Repeated capability: <repeatedCapabilityName>.
Repeated capability selector value: <selectorValue>.
```

Parameters

Inputs	Description	Base Type
`repeatedCapabilityName`	The name of the repeated capability (not the repeated capability instance).	`String`
`selectorValue`	The repeated capability selector value that contains the invalid range.	`String`

Usage

Since complex repeated capability selectors may not be used as indexers in IVI.NET, this exception should never be thrown by an indexer.

If the repeated capability selector does not support complex selectors, use the Invalid Selector exception.

If driver developers specify the message string, they are responsible for message string localization.

12.4.26 SimulationStateException

Description

After construction, the simulation property cannot be set to false, only to true. Some drivers may not allow simulation to be changed at all.

Exception

```
Ivi.Driver.SimulationStateException
```

Constructors

```
Ivi.Driver.SimulationStateException();

Ivi.Driver.SimulationStateException(String message);

Ivi.Driver.SimulationStateException(String message,
                                    System.Exception innerException);
```

Default Message String

```
The simulation state cannot be changed.
```

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.27 TriggerNotSoftwareException

Description

A Send Software Trigger method could not send a software trigger.

Exception

```
Ivi.Driver.TriggerNotSoftwareException(String message,
                                       String triggerSource)
```

Constructors

```
Ivi.Driver.TriggerNotSoftwareException();

Ivi.Driver.TriggerNotSoftwareException(String message);

Ivi.Driver.TriggerNotSoftwareException(String message,
                                       System.Exception innerException);
```

Default Message String

```
The trigger source is not set to software trigger.
Actual trigger source: <triggerSource>
```

Parameters

Inputs	Description	Base Type
`triggerSource`	The actual trigger source.	`String`

Usage

This exception should only be thrown by SendSoftwareTrigger() methods as defined in Section 2, *Software Triggering Capability*, of IVI-3.3, Cross Class Capability Specification.

If driver developers specify the message string, they are responsible for message string localization.

12.4.28 UnexpectedResponseException

Description

The driver received an unexpected response from the instrument.

Exception

```
Ivi.Driver.UnexpectedResponseException
```

Constructors

```
Ivi.Driver.UnexpectedResponseException();

Ivi.Driver.UnexpectedResponseException(String message);

Ivi.Driver.UnexpectedResponseException(String message,
                                       System.Exception innerException);
```

Default Message String

```
The response from the instrument was unexpected.
```

Usage

If driver developers specify the message string, they are responsible for message string localization.

12.4.29 UnknownOptionException

Description

The option string contains an option name that it does not recognize.

Exception

```
Ivi.Driver.UnknownOptionException
```

Constructors

```
Ivi.Driver.UnknownOptionException(String message,
                                  String optionName);

Ivi.Driver.UnknownOptionException();

Ivi.Driver.UnknownOptionException(String message);

Ivi.Driver.UnknownOptionException(String message,
                                  System.Exception innerException);
```

Default Message String

```
The option string contains an unknown option name.
Option name: <optionName>.
```

Parameters

Inputs	Description	Base Type
`optionName`	The unknown option name.	`String`

Usage

Since the driver is required to process option strings in the constructor, this exception shall only be thrown from the constructor.

If driver developers specify the message string, they are responsible for message string localization.

12.4.30 UnknownPhysicalNameException

Description

When establishing the map from virtual repeated capability names to physical repeated capability names, a physical name did not exist.

This exception also applies in cases where any member of a virtual range mapped to a physical name that did not exist.

Exception

```
Ivi.Driver.UnknownPhysicalNameException
```

Constructors

```
Ivi.Driver.UnknownPhysicalNameException(String driverSession,
                                        String repeatedCapabilityName,
                                        String virtualName,
                                        String physicalName);

Ivi.Driver.UnknownPhysicalNameException();

Ivi.Driver.UnknownPhysicalNameException(String message);

Ivi.Driver.UnknownPhysicalNameException(String message,
                                        System.Exception innerException);
```

Default Message String

```
The configuration store driver session references a physical name that is not
defined by the driver.
Driver session: <driverSession>
Repeated capability: <repeatedCapabilityName>
Virtual name: <virtualName>
Physical name: <physicalName>
```

Parameters

Inputs	Description	Base Type
`driverSession`	The name of the driver session in which the unknown physical name is referenced.	`String`
`repeatedCapabilityName`	The name of the repeated capability (not the repeated capability instance).	`String`
`virtualName`	The virtual name (defined for the repeated capability) which references the unknown physical name.	`String`
`physicalName`	The unknown physical name.	`String`

Usage

Since the driver is required to read all relevant configuration store information in the constructor, this exception shall only be thrown by the constructor.

If driver developers specify the message string, they are responsible for message string localization.

12.4.31 ValueNotSupportedException

Description

An enumerated value or a discrete value from a list of defined values is not supported by the specific driver.

Drivers should use Ivi.Driver.OutOfRangeException when they encounter other types of invalid values.

Exception

```
Ivi.Driver.ValueNotSupportedException
```

Constructors

```
Ivi.Driver.ValueNotSupportedException(String paramName,
                                      String value);

Ivi.Driver.ValueNotSupportedException();

Ivi.Driver.ValueNotSupportedException(String message);

Ivi.Driver.ValueNotSupportedException(String message,
                                      System.Exception innerException);
```

Default Message String

When instrument model is specified:

```
Value not supported.
Parameter name: <paramName>.
Value: <value>.
```

Parameters

Inputs	Description	Base Type
`paramName`	The name of the parameter to which the unsupported value is assigned.	`String`
`value`	The value that is not supported.	`String`

Usage

For a property set, the parameter name is “value”.

If driver developers specify the message string, they are responsible for message string localization.

12.5 IVI.NET Session Factory Method Exceptions

This section defines the list of IVI.NET session factory methods exceptions.

Refer to Section 2.9.2.2, *How Interchangeability Works in COM and .NET*, and Section 2.10, *The IVI Configuration Store*, of *IVI-3.1: Architecture Specifciation* for more information about IVI.NET session factory methods. Refer to *IVI-3.5: Configuration Server Specification*, for details regarding the information that is stored in the IVI configuration store.

The following exceptions defined by IVI.NET for use with session factory methods.

- ClassNameNotFoundException
- ConfigurationStoreLoadException
- DriverClassCreationException
- InvalidClassNameException
- SessionNotFoundException
- SoftwareModuleNotFoundException

Other exceptions may be thrown by session factory methods, but only if none of the exceptions listed above are applicable.

12.5.1 ClassNameNotFoundException

Description

The IVI.NET session factory method could not find the assembly qualified class name in the configuration store. Assembly qualified class name is a property of the IVI.NET specific driver's software module entry. It is needed to create an instance of the specific driver's main class.

This error is thrown after the driver session has been found, and the software module referenced by the driver session has been found. The cause of the error is either that the assembly qualified class name is blank, or that the program could not access the ISoftwareModule2 interface which contains the assembly qualified class name.

If the assembly qualified class name is blank, the driver referenced by the software module is not an IVI.NET driver, or the driver's software module entry is corrupt. If the driver is an IVI.NET driver, the problem may be fixed by repairing or reinstalling the driver.

If the assembly qualified class name is not blank, the program could not access the ISoftwareModule2 interface. This interface was added in version 1.5.0 of the IVI Shared Components. If a version of the IVI Shared Components older than version 1.5.0 is installed, the problem may be fixed by upgrading to a newer version.

Exception

```
Ivi.Driver.ClassNameNotFoundException
```

Constructors

```
Ivi.Driver.ClassNameNotFoundException(String driverSession,
                                      String softwareModule);

Ivi.Driver.ClassNameNotFoundException();

Ivi.Driver.ClassNameNotFoundException(String message);

Ivi.Driver.ClassNameNotFoundException(String message,
                                      System.Exception innerException);
```

Default Message String

```
IviSessionFactory: The specific driver’s main class (assembly qualified class
name) is not specified in the configuration store.
Driver session: <driverSession>
Specific driver (software module): <softwareModule>
```

Parameters

Inputs	Description	Base Type
`driverSession`	The name of the driver session to be instantiated by the IVI.NET session factory method.	`String`
`softwareModule`	The name of the specific driver's software module referenced by the driver session.	`String`

Usage

This exception shall only be thrown by IVI.NET session factory methods distributed by the IVI Foundation as part of the IVI.NET Shared Components.

12.5.2 ConfigurationStoreLoadException

Description

This exception is thrown when the session factory cannot load the configuration store file.

The IVI.NET session factory method could not load the configuration store specified.

This error is thrown if the driver cannot load the configuration store. The cause of the error is either that no file name can be found, the file specified does not exist, or the file could not be deserialized.

The session factory method gets the configuration store file name from configuration server's process default location, if it is specified, or from the master location, which should always reference an extant configuration store file.

If the file cannot be deserialized, then it does not conform to the version of the configuration store XML schema currently installed by the IVI Shared Components installer.

Exception

```
Ivi.Driver.ConfigurationStoreLoadException
```

Constructors

```
Ivi.Driver.ConfigurationStoreLoadException(System.Exception innerException);

Ivi.Driver.ConfigurationStoreLoadException();

Ivi.Driver.ConfigurationStoreLoadException(String message);

Ivi.Driver.ConfigurationStoreLoadException(String message,
                                           System.Exception innerException);
```

Default Message String

```
IviSessionFactory: There was an error loading the Configuration Server.
```

Usage

This exception shall only be thrown by IVI.NET session factory methods distributed by the IVI Foundation as part of the IVI.NET Shared Components.

12.5.3 DriverClassCreationException

Description

An instance of the specific driver referenced by the driver session name could not be created, or did not support the specified type.

This error is thrown after the driver session has been found, and the software module referenced by the driver session has been found. The cause of the error is that the specific driver could not be instantiated, or the driver did not support the sepcified type.

For example, the the session factory specifies that the type to be returned is IviDmm, but the driver doesn't support the IviDmm instrument class, this exception will be thrown.

Exception

```
Ivi.Driver.DriverClassCreationException
```

Constructors

```
Ivi.Driver.DriverClassCreationException(String softwareModule,
                                        String type,
                                        System.Exception exception);

Ivi.Driver.DriverClassCreationException(String softwareModule,
                                        String type);

Ivi.Driver.DriverClassCreationException();

Ivi.Driver.DriverClassCreationException(String message);

Ivi.Driver.DriverClassCreationException(String message,
                                        System.Exception innerException);
```

Default Message String

```
IviSessionFactory: An instance of the specific driver referenced by the logical
name or driver session name could not be created, or did not support the
specified type.
Specific driver (software module): <softwareModuleName>
Type: <type>
```

Parameters

Inputs	Description	Base Type
`softwareModule`	The name of the specific driver's software module.	`String`
`type`	The type, supported by the specific driver, that the session factory method is trying to return.	`String`

Usage

This exception shall only be thrown by IVI.NET session factory methods distributed by the IVI Foundation as part of the IVI.NET Shared Components.

12.5.4 InvalidClassNameException

Description

This exception thrown when the session facotry determines that the specific driver's main class (assembly qualified class name) is not formatted properly in the configuration store.

The IVI.NET session factory method has determined that the specific driver's main class name (assembly qualified class name) is not formatted properly in the configuration store. The format is "FullAssemblyName;NamespaceQualifiedTypeName".

This error is thrown after the driver session has been found, and the software module referenced by the driver session has been found. The cause of the error is that the assembly qualified class name is not correctly formatted.

Example of a correctly formatted assembly qualified class name:
"Ivi.Driver.dll, Version=1.0.0.0, Culture=neutral, PublicKeyToken=a128c98f1d7717c1, processorArchitecture=MSIL"

Exception

```
Ivi.Driver.InvalidClassNameException
```

Constructors

```
Ivi.Driver.InvalidClassNameException(String softwareModule,
                                     String assemblyQualifiedClassName);

Ivi.Driver.InvalidClassNameException();

Ivi.Driver.InvalidClassNameException(String message);

Ivi.Driver.InvalidClassNameException(String message,
                                     System.Exception innerException);
```

Default Message String

```
IviSessionFactory: The IVI.NET driver's assembly qualified class name is not
formatted correctly in the configuration store.  The correct format is
"FullAssemblyName;NamespaceQualifiedTypeName".
Specific driver (software module): <softwareModule>
Assembly qualified class name: <assemblyQualifiedClassName>
```

Parameters

Inputs	Description	Base Type
`softwareModule`	The name of the specific driver's software module.	`String`
`assemblyQualifiedClassName`	The driver's Assembly Qualified Class Name.	`String`

Usage

This exception shall only be thrown by IVI.NET session factory methods distributed by the IVI Foundation as part of the IVI.NET Shared Components.

12.5.5 SessionNotFoundException

Description

The IVI.NET session factory method could not find a driver session that could be used to instantiate an IVI.NET instrument driver.

Name may refer to either a logical name or a physical name in the configuration store. This error is thrown if the session factory method cannot find either a logical name or a driver session name that matches the specified name, or the logical name references a driver session that cannot be found.

Exception

```
Ivi.Driver.SessionNotFoundException
```

Constructors

```
Ivi.Driver.SessionNotFoundException(String message,
                                    String name);

Ivi.Driver.SessionNotFoundException();

Ivi.Driver.SessionNotFoundException(String message);

Ivi.Driver.SessionNotFoundException(String message,
                                    System.Exception innerException);
```

Default Message String

```
IviSessionFactory: The driver session referenced by the specified Logical Name
or Driver Session Name does not exist in the configuration store.
Name: <name>
```

Parameters

Inputs	Description	Base Type
`name`	Name may refer to either a logical name or a physical name in the configuration store.	`String`

Usage

This exception shall only be thrown by IVI.NET session factory methods distributed by the IVI Foundation as part of the IVI.NET Shared Components.

12.5.6 SoftwareModuleNotFoundException

Description

The IVI.NET session factory method could not find the software module referenced by the driver session in the configuration store.

This error is thrown after the driver session has been found. The cause of the error is that the software module referenced by the driver session could not be found.

In some cases, a driver session is connected to a specific driver's software module, and then that driver is uninstalled, removing the software module entry from the configuration store. In this case, the software module reference is maintined in the driver session, but the software module itself is missing. This can be addressed by reinstalling the driver.

Exception

```
Ivi.Driver.SoftwareModuleNotFoundException
```

Constructors

```
Ivi.Driver.SoftwareModuleNotFoundException(String driverSession,
                                           String softwareModule);

Ivi.Driver.SoftwareModuleNotFoundException();

Ivi.Driver.SoftwareModuleNotFoundException(String message);

Ivi.Driver.SoftwareModuleNotFoundException(String message,
                                           System.Exception innerException);
```

Default Message String

```
IviSessionFactory: The IVI.NET specific driver software module referenced by
the driver session does not exist in the configuration store.
Driver session: <driverSession>
Specific driver (software module): <softwareModuleName>
```

Parameters

Inputs	Description	Base Type
`driverSession`	The name of the driver session to be instantiated by the IVI.NET session factory method.	`String`
`softwareModule`	The name of the specific driver's software module referenced by the driver session.	`String`

Usage

This exception shall only be thrown by IVI.NET session factory methods distributed by the IVI Foundation as part of the IVI.NET Shared Components.

12.6 Warnings

Table 12-3 lists the standard IVI.NET warnings that IVI driver developers may use during driver development. It also specifies the recommended format of the error description string for those error codes.

In the messages listed below, {0} must include the name of the method or property responsible for the warning.

Table 12-3. IVI.NET Warnings

Name	Warning GUID
ID Query Not Supported	"37FC4913-27D4-4dee-90FC-87CED0677D72"
	"{0}: ID Query is not supported by this instrument."
Self Test Not Supported	"32B87F50-501E-4c95-A782-FBEECF7FB324"
	"{0}: Self test is not supported by this instrument."
Error Query Not Supported	"BE37BF5D-FAE5-44d0-8AA4-4B521D1D17DE"
	"{0}: Error query is not supported by this instrument."
Revision Query Not Supported	"278665CA-DCCC-49ad-A76C-3B963143DD20"
	"{0}: Firmware revision query is not supported by this instrument."