IVI

Interchangeable
Virtual
Instruments

IVI-3.9: C Shared Components Specification

April 2008 Edition
Revision 1.0

Important Information

The C Shared Components 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`.

Table of Contents

IVI C Shared Components Specification

Revision History

This section is an overview of the revision history of the C Shared Components 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. C Shared Components Specification

Revision Number	Date of Revision	Revision Notes
Revision 0.1	November 06, 2000	Original draft.
Revision 0.9	February 01, 2001	Reformatted the document and implemented changes according to the November meeting minutes.
Revision 1.0	June 27, 2001	Final draft ready for review
Revision 1.0	February 18, 2008	Editorial changes to Sections 4.7 and 5.4
Revision 1.0	April 29, 2008	Editorial change to update the IVI Foundation contact information in the Important Information section to remove obsolete address information and refer only to the IVI Foundation web site.

1. Summary of Contents

The IVI C shared components are intended to aid in the development of IVI-C drivers.

Several of the shared components are intended to be used only by other shared components. However, their interfaces are defined in this document so that drivers can make use of them directly if necessary.

The following table summarizes the intended users of each component.

Table 1-1. C Shared Components and Intended Users

Component	**Intended user(s)**
Dynamic Driver Loader	Class Driver
Error Message	Specific Driver, Class Driver
Session Management	Specific Driver, Class Driver
Session Error	Session Management Component
Multithread Lock	Session Management Component
Thread Local Error Storage	Session Management Component
Thread Local Storage	Thread Local Error Storage Component

1.1 References

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

- IVI 3.2—Inherent Capabilities Specification

1.2 Implementation

The current installation package for the IVI Foundation Common Components, including the IVI C Shared Components, is available from the IVI Foundation website at http://www.ivifoundation.org.

2. Dynamic Driver Loader

An IVI class driver uses this component to dynamically load an IVI-C class-compliant specific driver and obtain pointers to the class-defined functions that the IVI-C class-compliant specific driver exports. Before calling the functions in this component, the IVI class driver obtains the path and prefix for the IVI-C class compliant specific driver module associated with a logical name from the IVI configuration server.

The Dynamic Driver Loader shared component can load the following different types of driver modules.

Dynamic Libraries—Under Windows 2000/NT/Me/9*x*, the component can load a dynamic link library (`.dll`) file. Under Sun Solaris/HP-UX/Linux, the component can load a shared object (`.so`) file.

Static Libraries—If the LabWindows/CVI Run Time Engine is present on the system, the component can load a static library. Under Windows 2000/NT/Me/9*x*, the component can load an object (`.obj`) file or a static library (`.lib`) file. Under Sun Solaris, the component can load an object (`.o`) or static library (`.a`) file.

Source Code—If the application is running in the LabWindows/CVI development environment, the component can load a source (`.c`) file under Windows 2000/NT/Me/9*x* and Sun Solaris, provided the source code file is included in the LabWindows/CVI project.
The following table summarizes what module types the Dynamic Driver Loader component can load under various operating systems.

Table 2-1. Module Types Loadable in Various Operating Systems

Operating System	Dynamic Library	Static Library (LabWindows/CVI RTE)	Source File (LabWindows/CVI Environment)
32-bit Windows	Yes	Yes	Yes
Sun Solaris	Yes	Yes	Yes
Linux	Yes	No	No
HP-UX	Yes	No	No

The Dynamic Driver Loader component defines the following functions:

- IviDriverLoader_New
- IviDriverLoader_GetFunctionPtr
- IviDriverLoader_GetFunctionPtrByName
- IviDriverLoader_Dispose

The Dynamic Driver Loader component defines the following type:

```
typedef struct IviDriverLoaderStruct *IviDriverLoader;
```

2.1 New

Description

This function creates an object that dynamically loads the IVI-C class-compliant specific driver that is associated with the path and prefix specified by the caller. The caller obtains the addresses of functions through the `IviDriverLoader_GetFunctionPtr` function.

If the object cannot load the driver module, this function returns an error.

C Prototype

```
ViStatus IviDriverLoader_New (ViConstString Path,
                              ViConstString Prefix,
                              IviDriverLoader *Handle);
```

Parameters

Input	Description	Data Type
`Path`	The pathname of the IVI-C class-compliant specific driver module.	`ViConstString`
`Prefix`	The prefix of the C functions that the IVI-C class-compliant specific driver exports.	`ViConstString`

Output	Description	Data Type
`Handle`	The handle to the object that this function creates.	`IviDriverLoader`

2.2 Get Function Pointer

Description

This function returns the address of a function in the IVI-C class-compliant specific driver module. The caller specifies the function by name without the specific driver prefix.

Caution: The function does not validate the `Handle` parameter.

If function is unable to obtain the address of the requested function, this function returns `VI_NULL` as the function address. It does not return an error in this case.

C Prototype

```
ViStatus IviDriverLoader_GetFunctionPtr(IviDriverLoader Handle,
                                        ViConstString FunctionName,
                                        ViAddr *FunctionPtr);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to the object that `IviDriverLoader_New` creates.	`IviDriverLoader`

`FunctionName`	Function name, without specific driver prefix, for which caller wants to retrieve a function pointer.	`ViConstString`

Output	Description	Data Type
`FunctionPtr`	The address of the function that `FunctionName` specifies. The value is `VI_NULL` if the IVI-C class-compliant specific driver module does not export the function.	`ViAddr`

Return Values

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

2.3 Get Function Pointer By Complete Name

Description

This function returns the address of a function in the IVI-C class-compliant specific driver module. The caller specifies the function by its complete name, including the instrument driver prefix.

Caution: The function does not validate the `Handle` parameter.

If function is unable to obtain the address of the requested function, this function returns `VI_NULL` as the function address. It does not return an error in this case.

C Prototype

```
ViStatus IviDriverLoader_GetFunctionPtrByCompleteName(
                                                      IviDriverLoader Handle,
                                                      ViConstString FunctionName,
                                                      ViAddr *FunctionPtr);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to the object that `IviDriverLoader_New` creates.	`IviDriverLoader`
`FunctionName`	Complete function name for which caller wants to retrieve a function pointer.	`ViConstString`

Output	Description	Data Type
`FunctionPtr`	The address of the function that `FunctionName` specifies. The value is `VI_NULL` if the IVI-C class-compliant specific driver module does not export the function.	`ViAddr`

Return Values

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

2.4 Dispose

Description

This function unloads the IVI-C class-compliant specific driver and destroys an object created by `IviDriverLoader_New`.

Caution: The function does not validate the `Handle` parameter.

C Prototype

```
void IviDriverLoader_Dispose (IviDriverLoader Handle);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to the object that `IviDriverLoader_New` creates.	`IviDriverLoader`

Return Values

None

3. Error Message

The purpose of this component is to help driver developers create error messages. The component defines the following data type.

```
#ifdef WIN32
    #pragma pack(push)
    #pragma pack(4)
#endif

typedef struct
{
   ViStatus errorCode;
   ViConstString errorMessage;
}IviErrorTableEntry, *IviErrorTable;

#ifdef WIN32
    #pragma pack(pop)
#endif
```

The Error Message component defines the following functions:

- IviErrorMessage_Get
- IviErrorMessage_FormatWithElaboration

3.1 Get

Description

This function retrieves the static message associated with a specific error code. The error code must be one of the following.

- An error code that one or more of the C shared components defines
- An error code that the IVI foundation defines in *IVI-3.2*: *Inherent Capabilities Specification*
- An error code in the table passed to the `ErrorTable` parameter.

An IVI-C driver can use the `ErrorTable` parameter to pass a list of the driver's specific error codes, as well as any class-defined error codes that it might return.

If the function cannot find the error message for the error code, the function returns a pointer to an empty string in the `ErrorMessage` parameter and returns an error code.

The caller should not deallocate the error message string.

C Prototype

```
ViStatus IviErrorMessage_Get (ViStatus ErrorCode,
                              IviErrorTable ErrorTable,
                              ViConstString *ErrorMessage);
```

Parameters

Input	Description	Data Type
`ErrorCode`	Error code value.	`ViStatus`

`ErrorTable`	A list of error codes and associated error messages. The last element of the table has `VI_SUCCESS` for the errorCode element and an empty string for the `errorMessage` element. If the parameter is `VI_NULL`, it is ignored.	`IviErrorTable`

Output	Description	Data Type
`ErrorMessage`	The string that contains the error message that corresponds to the `ErrorCode`. The caller should not deallocate this string.	`ViConstString`

Return Values

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

3.2 Format With Elaboration

Description

This function formats an error description from two error messages and places it into an output buffer.

Refer to Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters of the IVI–3.2: Inherent Capabilities Specification* for rules regarding the `ErrorDescriptionBufferSize` and `ErrorDescription` parameters.

By passing zero 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.

C Prototype

```
ViStatus IviErrorMessage_FormatWithElaboration (ViConstString ErrorMessage,
                                                ViConstString ErrorElaboration,
                                                ViInt32 ErrorDescriptionBufferSize,
                                                ViChar ErrorDescription[ ]);
```

Parameters

Input	Description	Data Type
`ErrorMessage`	The more general error message. Normally, this is a static error message associated with an error code.	`ViConstString`
`ErrorElaboration`	The more specific error message that refines the more general error message.	`ViConstString`
`ErrorDescriptionBufferSize`	The number of bytes in the `ViChar` array that the caller specifies for the `ErrorDescription` parameter.	`ViInt32`

Output	Description	Data Type
`ErrorDescription`	Buffer into which the function places the formatted error description. The buffer shall contain at least as many bytes as the caller specifies in the `ErrorDescrptionBufferSize` parameter. The caller may pass `VI_NULL` for this parameter if the `ErrorDescrptionBufferSize` parameter is zero .	`ViChar []`

Return Values

None.

4. Session Management

IVI-C drivers use this component to create and destroy IVI driver sessions, associate instance data with sessions, and obtain multithread locks on sessions.

This component contains error handling functions that parallel the Get Error and Clear Error functions defined in the *IVI 3.2: Inherent Capabilities Specification*. This component also contains a corresponding Set Error function. These functions set, get, and clear the error information for an IVI session and for the current execution thread.

The Set Error function operates in a manner that is consistent with the behavior of the Get Error function specified in the *IVI 3.2: Inherent Capabilities Specification*. In particular, the Set Error function does not overwrite the existing status code unless the severity of the status code is greater than the existing one.

Typically, an IVI-C driver can handle the recording and retrieval of all error information entirely through calls to the functions in this component.

All functions that take `ViSession` handle as an input parameter are multithread safe. Threads that attempt to call functions are blocked if another thread is currently calling a function in the same component using the same `ViSession` handle.

The Session Management component defines the following functions:

- IviSession_New
- IviSession_SetDataPtr
- IviSession_GetDataPtr
- IviSession_Lock
- IviSession_Unlock
- IviSession_SetError
- IviSession_GetError
- IviSession_ClearError
- IviSession_Dispose

4.1 New

Description

This function creates an IVI driver session.

C Prototype

```
ViStatus IviSession_New (ViSession *Handle);
```

Parameters

Output	Description	Data Type
`Handle`	Handle to the IVI driver session that this function creates.	`ViSession`

Return Values

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

4.2 Set Data Pointer

Description

This function associates a pointer value with a session handle. An IVI-C driver uses this function to store a pointer to its session instance data.

C Prototype

```
ViStatus IviSession_SetDataPtr (ViSession Handle, ViAddr DataPtr);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`
`DataPtr`	A pointer to dynamically allocated data that the caller wants to associate with the session.	`ViAddr`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

4.3 Get Data Pointer

Description

This function retrieves the pointer value associated with the session handle.

C Prototype

```
ViStatus IviSession_GetDataPtr (ViSession Handle, ViAddr *DataPtr);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`

Output	Description	Data Type
`DataPtr`	The pointer value associated with the session in the most recent call to `IviSession_SetDataPtr`. The value is `VI_NULL` if no pointer value is associated with the session.	`ViAddr`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

4.4 Lock

Description

This function obtains a multithread lock on an IVI driver session. The function waits for all other execution threads to release their locks on the session before it proceeds.

The caller can safely make nested calls to this function within the same thread. To completely unlock the session, the caller must balance each call to this function with a call to `IviSession_Unlock`, unless the caller uses the `HasLock` parameter.

The `HasLock` parameter is available for the convenience of the caller. It relieves the caller from the burden of having to keep track of how many times a function locks and unlocks a session. To use this parameter effectively:

- The caller declares a `ViBoolean` local variable and initializes it to `VI_FALSE`,
- Passes it by reference to each call to `IviSession_Lock` within the function,
- Has no early `return` statements in the function, and
- Calls `IviSession_Unlock` just once at the end of the function.

If the caller does not want to use the `HasLock` parameter, the caller passes `VI_NULL`.

If the `HasLock` parameter points to a value of `VI_TRUE`, `IviSession_Lock` does nothing. If the `HasLock` parameter points to a value of `VI_FALSE`, `IviSession_Lock` obtains a lock on the session and sets `HasLock` to point to `VI_TRUE`.

C Prototype

```
ViStatus IviSession_Lock (ViSession Handle, ViBoolean *HasLock);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`

Input/Output	Description	Data Type
`HasLock`	A reference to a local variable in the calling function that indicates if the calling function currently has a lock on the session. The caller may pass `VI_NULL` for this parameter.	`ViBoolean`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

4.5 Unlock

Description

This function releases a multithread lock that the caller previously acquired on an IVI driver session. Refer to `IviSession_Lock` for more information on IVI session locks.

If the `HasLock` parameter points to a value of `VI_TRUE`, `IviSession_Unlock` unlocks the session and sets the value `HasLock` points to `VI_FALSE`. If the `HasLock` parameter points to a value of `VI_FALSE`, `IviSession_Unlock` does nothing.

The caller may pas `VI_NULL` for the `HasLock` parameter. If the calling function does not have the lock on the session, the behavior is undefined.

C Prototype

```
ViStatus IviSession_Unlock (ViSession Handle, ViBoolean *HasLock);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`

Input/Output	Description	Data Type
`HasLock`	A reference to a local variable in the calling function that indicates if the calling function currently has a lock on the session. The caller may pass `VI_NULL` for this parameter.	`ViBoolean`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

4.6 Set Error

Description

This function sets the error information for an IVI session and for the current execution thread. If the caller passes a valid IVI session for the `Handle` parameter, this function sets the error information for the session and the error information for the current execution thread. If the caller passes `VI_NULL` for the `Handle` parameter, the function sets the error information for the current execution thread only.

This function operates in a manner that is consistent with the behavior of the Set Error function specified in the *IVI 3.2: Inherent Capabilities Specification*. In particular, the function does not overwrite the existing error code unless the severity of the error code is greater than the existing error. Error codes have three severity levels. Negative error codes are at the highest level. Positive error codes constitute warnings and are at the second level. A zero error code (Success) is at lowest level. The following table describes the exact behavior of this function in setting the error code for the session and for the current execution thread.

Table 4-1. Exact Behavior in Setting Error Codes

ErrorCode Parameter	Existing Error Code	Update Error Code?
Negative value	Negative value	No
	Positive value	Yes
	Zero	Yes
Positive value	Negative value	No
	Positive value	No
	Zero	Yes
Zero	Negative value	No
	Positive value	No
	Zero	No

The function overwrites the existing error description if either of the following conditions are met:

- The function overwrites the existing error code or the `ErrorCode` parameter is equal to the existing error code, and
- The existing error description is `VI_NULL` or an empty string.

When the function overwrites the existing error description and the `ErrorDescription` parameter is `VI_NULL` or an empty string, the function stores `VI_NULL` for the error description. When the function overwrites the existing error description and the `ErrorDescription` parameter is a non-empty string, the function allocates a copy of the `ErrorDescription` parameter.

The function deallocates the existing description string before overwriting it.

If the function encounters an error, it stores as much of the information from the parameters as it can before returning an appropriate error code. For example, if the caller passes an invalid handle for the session, the function sets the error information for the current execution thread and returns an `IVI_ERROR_INVALID_SESSION_HANDLE` error. If the function runs out of memory while attempting to allocate a copy of the error description string, the function stores the value of the `ErrorCode` parameter, stores `VI_NULL` for the error description, and returns an `IVI_ERROR_OUT_OF_MEMORY` error.

C Prototype

```
ViStatus IviSession_SetError (ViSession Handle, ViStatus ErrorCode,
                              ViConstString ErrorDescription);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session that the caller creates with `IviSession_New`. The caller may pass `VI_NULL`, in which case the function sets the error information only for the current execution thread.	`ViSession`
`ErrorCode`	Error code to store in the session.	`ViStatus`
`ErrorDescription`	Fully formatted error description string to store in the session. The caller may pass `VI_NULL` to indicate an empty string.	`ViConstString`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

4.7 Get Error

Description

This function retrieves and then clears the IVI error information for the session or the current execution thread. It does so in a manner consistent with the Get Error function defined in the *IVI–3.2: Inherent Capabilities Specification*. If the `Handle` parameter is a valid IVI session, the function retrieves and then clears the error information for the session. If the `Handle` parameter is `VI_NULL`, the function retrieves and then clears the error information for the current execution thread. If the `Handle` parameter is an invalid session, the function does nothing and returns the Invalid Session Handle error.

The function returns the current error code in the `ErrorCode` parameter and copies the current error description string into the `ErrorDescription` parameter. Refer to Section 3.1.2.1, *Additional Compliance Rules for C Functions with ViChar Array Output Parameters of the IVI–3.2: Inherent Capabilities Specification* for rules regarding the `ErrorDescriptionBufferSize` and `ErrorDescription` parameters.

After retrieving the error code and error description, the function clears the error information, except in one case. If the `ErrorDescriptionBufferSize` parameter is zero, the function does not clear the error information. By passing zero 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.

C Prototype

```
ViStatus IviSession_GetError (ViSession Handle,
                              ViInt32 ErrorDescriptionBufferSize ,
                              ViStatus *ErrorCode,
                              ViChar ErrorDescription[ ])
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session that the caller creates with `IviSession_New`. The caller may pass `VI_NULL` to request the error information for the current execution thread.	`ViSession`
`ErrorDescriptionBufferSize`	The number of bytes in the `ViChar` array that the caller specifies for the `ErrorDescription` parameter.	`ViInt32`

Output	Description	Data Type
`ErrorCode`	Returns the current error code. Zero indicates that no error occurred. A positive value indicates a warning. A negative value indicates an error. The caller may pass `VI_NULL` for this parameter if the caller does not want to retrieve this value.	`ViStatus`
`ErrorDescription`	Buffer into which the function copies the error description string. The buffer shall contain at least as many bytes as the caller specifies in the `ErrorDescrptionBufferSize` parameter. The caller may pass `VI_NULL` for this parameter if the `ErrorDescrptionBufferSize` parameter is zero .	`ViChar [ ]`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

4.8 Clear Error

Description

This function clears the error information for an IVI session or for the current execution thread. If the caller passes a valid IVI session for the `Handle` parameter, this function clears the error information for the session. If caller passes `VI_NULL` for the `Handle` parameter, the function clears the error information for the current execution thread. If the `Handle` parameter is an invalid session, the function does nothing and returns the Invalid Session Handle error.

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

C Prototype

```
ViStatus IviSession_ClearError (ViSession Handle);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session that the caller creates with `IviSession_New`. The caller may pass `VI_NULL`, to clear the error information for the current execution thread.	`ViSession`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

4.9 Dispose

Description

This function closes the instrument driver session.

This function does not free the session instance data that the caller associates with the session handle by calling `IviSession_SetDataPtr`. The caller must explicitly deallocate the session instance data.

C Prototype

```
ViStatus IviSession_Dispose (ViSession Handle)
```

Parameters

Input	Description	Data Type
`Handle`	Handle to the IVI driver session.	`ViSession`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Invalid Session Handle	`IVI_ERROR_INVALID_SESSION_HANDLE`

5. Session Error

IVI-C drivers maintain error information on both a session basis and a thread-local basis. All calls to IVI-C driver functions in the same thread share the same thread-local error information. This component contains low-level functions to access the error code and error description string for a session.

The Session Management component uses the functions in this component. Typically, IVI-C drivers do not call these functions directly.

The Session Error component defines the following functions:

- IviSessionError_SetErrorCode
- IviSessionError_GetErrorCode
- IviSessionError_SetErrorDescription
- IviSessionError_GetErrorDescription

5.1 Set Error Code

Description

This function sets the error code for an IVI session.

Caution: The function does not validate the `Handle` parameter.

C Prototype

```
ViStatus IviSessionError_SetErrorCode (ViSession Handle,
                                       ViStatus ErrorCode);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`
`ErrorCode`	Error code to store in the session error information.	`ViStatus`

Return Values

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

5.2 Get Error Code

Description

This function retrieves the error code for an IVI session.

Caution: The function does not validate the `Handle` parameter.

C Prototype

```
ViStatus IviSessionError_GetErrorCode (ViSession Handle,
                                       ViStatus *ErrorCode);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`

Output	Description	Data Type
`ErrorCode`	The current value of the error code in the session error information.	`ViStatus`

Return Values

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

5.3 Set Error Description

Description

This function sets the error description string for an IVI session.

Caution: The function does not validate the `Handle` parameter.

The function deallocates the existing error description string for the session. The function allocates a copy of the `ErrorDescription` string parameter and stores the address of the copy in the session. If the `ErrorDescription` parameter is `VI_NULL` or an empty string, the function stores `VI_NULL` for the address.

C Prototype

```
ViStatus IviSessionError_SetErrorDescription (ViSession Handle,
ViConstString ErrorDescription);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`
`ErrorDescription`	Fully formatted error description string to store in the session error information. The caller may pass `VI_NULL` to indicate an empty string.	`ViConstString`

Return Values

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

5.4 Get Error Description

Description

This function retrieves the address of the error description string for an IVI session. A value of `VI_NULL` for the address indicates an empty string.

Caution: The function does not validate the `Handle` parameter.

The caller should not deallocate the error description string.

C Prototype

```
ViStatus IviSessionError_GetErrorDescription (ViSession Handle,
                                              ViConstString *ErrorDescription);
```

Parameters

Input	Description	Data Type
`Handle`	Handle to an IVI driver session.	`ViSession`

Output	Description	Data Type
`ErrorDescription`	Address of the current error description string in the session error information. The result value can be `VI_NULL`.	`ViConstString`

Return Values

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

6. Multithread Lock

This component provides functions that make it possible to use multithread locks in an operating-system-independent manner. The Session Management component uses the functions in this component.

It is safe to use this component on operating systems that do not support multiple threads.

The Multithread Lock component defines the following functions:

- IviMultithreadLock_New
- IviMultithreadLock_Acquire
- IviMultithreadLock_Release
- IviMultithreadLock_Dispose

This component defines the following type:

```
typedef struct IviMultithreadLockStruct *IviMultithreadLock;
```

6.1 New

Description

This function creates a multithread lock.

C Prototype

```
ViStatus IviMultithreadLock_New (IviMultithreadLock *Lock);
```

Parameters

Output	Description	Data Type
`Lock`	Handle to the multithread lock that this function creates.	`IviMultithreadLock`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Could Not Create Lock	`IVI_ERROR_CANNOT_CREATE_LOCK`

6.2 Acquire

Description

This function acquires a multithread lock. It waits for all other execution threads to release the lock before it proceeds. Every call to this function must be balanced by a call to the `IviMultithreadLock_Release` function.

Caution: The function does not validate the `Lock` parameter.

C Prototype

```
void IviMultithreadLock_Acquire (IviMultithreadLock Lock);
```

Parameters

Input	Description	Data Type
`Lock`	Handle to a multithread lock.	`IviMultithreadLock`

Return Values

None

6.3 Release

Description

This function releases a multithread lock. Every call to the `IviMultithreadLock_Acquire` function must be balanced by a call to this function.

Caution: The function does not validate the `Lock` parameter.

C Prototype

```
void IviMultithreadLock_Release (IviMultithreadLock Lock);
```

Parameters

Input	Description	Data Type
`Lock`	Handle to a multithread lock.	`IviMultithreadLock`

Return Values

None

6.4 Dispose

Description

This function destroys a multithread lock.

Caution: The function does not validate the `Lock` parameter.

C Prototype

```
void IviMultithreadLock_Dispose (IviMultithreadLock Lock);
```

Parameters

Input/Output	Description	Data Type
`Lock`	Handle to a multithread lock.	`IviMultithreadLock`

Return Values

None

7. Thread-Local Error Storage

IVI-C drivers maintain error information on both a session basis and a thread-local basis. All calls to IVI-C driver functions in the same thread share the same thread-local error information. This component contains low-level functions to access the error code and error description string for the current execution thread. The component stores the error information in a dynamically allocated data structure pointed to by a thread-local variable.

The Session Management component uses the functions in this component. Typically, IVI-C drivers do not call these functions directly.

It is safe to use this component on operating systems that do not have thread-local variables.

The Thread-Local Error Storage component defines the following functions:

- IviThreadError_SetErrorCode
- IviThreadError_GetErrorCode
- IviThreadError_SetErrorDescription
- IviThreadError_GetErrorDescription

7.1 Set Error Code

Description

This function sets the IVI error code for the current execution thread.

C Prototype

```
ViStatus IviThreadError_SetErrorCode (ViStatus ErrorCode);
```

Parameters

Input	Description	Data Type
`ErrorCode`	Error code to store in the thread-local error information.	`ViStatus`

Return Values

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

7.2 Get Error Code

Description

This function retrieves the IVI error code for the current execution thread.

If the error code for the thread-local error information has not yet been set, the function returns `IVI_SUCCESS` in the `ErrorCode` parameter. If the error code has been set, the function returns the thread-local error code in the `ErrorCode` parameter.

C Prototype

```
ViStatus IviThreadError_GetErrorCode (ViStatus *ErrorCode);
```

Parameters

Output	Description	Data Type
`ErrorCode`	The current value of the thread-local error code.	`ViStatus`

Return Values

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

7.3 Set Error Description

Description

This function sets the IVI error description for the current execution thread.

The function deallocates the existing error description string for the thread. The function allocates a copy of the `ErrorDescription` string parameter and stores the address of the copy in the thread. If the `ErrorDescription` parameter is `VI_NULL` or an empty string, the function stores `VI_NULL` for the address.

C Prototype

```
ViStatus IviThreadError_SetErrorDescription (ViConstString ErrorDescription);
```

Parameters

Input	Description	Data Type
`ErrorDescription`	Fully formatted error description string to store in the thread-local error information. The caller may pass `VI_NULL` to indicate an empty string.	`ViConstString`

Return Values

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

7.4 Get Error Description

Description

This function retrieves the address of the IVI error description string for the current execution thread.

If the error description for the thread-local error information has not yet been set, the function returns `VI_NULL` in the `ErrorDescription` parameter. If the error code has been set, the function returns the address of the thread-local error description string in the `ErrorDescription` parameter. A value of `VI_NULL` indicates an empty string.

The caller should not deallocate the error description string.

C Prototype

```
ViStatus IviThreadError_GetErrorDescription (ViConstString *ErrorDescription);
```

Parameters

Output	Description	Data Type
`ErrorDescription`	Address of the current error description string in the thread-local error information. Can be `VI_NULL`.	`ViConstString`

Return Values

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

8. Thread-Local Storage

This component provides functions that make it possible to store and retrieve a thread-local data structure in an operating-system-independent manner. The Thread-Local Error Storage component uses the functions in this component. Typically, IVI-C drivers do not call these functions directly.

It is safe to use this component on operating systems that do not have thread-local variables.

On some operating systems, the number of thread-local variables is severely limited. Consequently, the suggested way to use thread-local variables is to define a data structure that contains all the information that you want to store on a thread-local basis, allocate an instance of the data structure in each thread, and store the address of each instance in a thread-local variable. This component allows for storing such an address in a thread-local variable.

The Thread-Local Storage component defines the following functions:

- IviThreadVar_New
- IviThreadVar_SetValueViAddr
- IviThreadVar_GetValueViAddr
- IviThreadVar_Dispose

This component defines the following type:

```
typedef struct IviThreadVarStruct* IviThreadVar;
typedef void (*IviThreadVarFreeFuncPtr) (ViAddr ptr);
```

8.1 New

Description

This function initializes a thread-local variable. On Windows operating systems, the typical place to call this function is in the `WinMain` or `DllMain` function in response to the `DLL_PROCESS_ATTACH` event. On Posix operating systems, this function iscalled in a function that is declared with the `init` pragma.

Posix operating systems invoke the function pointed to by the `FreeFn` parameter whenever a thread goes out of existence.

On Windows operating systems, the caller invokes the `FreeFn` function in response to the `DLL_PROCESS_DETACH` and `DLL_THREAD_DETACH` events in the `WinMain` or `DllMain` function. The code for the `DLL_PROCESS_DETACH` event invokes the `FreeFn` function before calling `IviThreadVar_Dispose`.

C Prototype

```
ViStatus IviThreadVar_New (IviThreadVarFreeFuncPtr FreeFn,
                           IviThreadVar *ThreadVar);
```

Parameters

Input	Description	Data Type
`FreeFn`	Pointer to a function that can deallocate the data structure that the caller stores in the thread-local variable.	`IviThreadVarFreeFuncPtr`

Output	Description	Data Type
`ThreadVar`	Handle to the thread-local variable that this function creates.	`IviThreadVar`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. The table below specifies additional status codes for this function.

Name	C Identifier
Could Not Create Thread Local	`IVI_ERROR_CANNOT_CREATE_THREAD_LOCAL`

8.2 Set Value ViAddr

Description

This function stores an address in a thread-local variable for the current execution thread.

Caution: The function does not validate the `ThreadVar` parameter.

C Prototype

```
void IviThreadVar_SetValueViAddr (IviThreadVar ThreadVar, ViAddr Val)
```

Parameters

Input	Description	Data Type
`ThreadVar`	Handle to a thread-local variable.	`IviThreadVar`
`Val`	Address to store in the thread-local variable.	`ViAddr`

Return Values

None

8.3 Get Value ViAddr

Description

This function retrieves the address stored in a thread-local variable for the currently executing thread.

Caution: The function does not validate the `ThreadVar` parameter.

C Prototype

```
void IviThreadVar_GetValueViAddr (IviThreadVar ThreadVar, ViAddr *Val)
```

Parameters

Input	Description	Data Type
`ThreadVar`	Handle to a thread-local variable.	`IviThreadVar`

Output	Description	Data Type
`Val`	Address currently stored in the thread-local variable.	`ViAddr`

Return Values

None

8.4 Dispose

Description

This function destroys a thread-local variable. On Windows operating systems, the typical place to call this function is in the `WinMain` or `DllMain` function in response to the `DLL_PROCESS_DETACH` event. On Posix operating systems, this function is called in a function that is declared with the `fini` pragma.

Caution: The function does not validate the `ThreadVar` parameter.

C Prototype

```
void IviThreadVar_Dispose (IviThreadVar ThreadVar);
```

Parameters

Input	Description	Data Type
`ThreadVar`	Handle to a thread-local variable.	`IviThreadVar`

Return Values

None

9. Error and Completion Codes

The following table specifies the actual values and static messages for the Error and Completion codes defined for the IVI-C Shared Components.

Table 9-1. IVI-C Error and Completion Code Values

Actual Value	C Completion Code	Eror Message
`IVI_SHARED_COMPONENT_ERROR_BASE + 0x190`	`IVI_ERROR_INVALID_SESSION_HANDLE`	The session handle is not valid.
`IVI_SHARED_COMPONENT_ERROR_BASE + 0x198`	`IVI_ERROR_CANNOT_CREATE_LOCK`	Could not create a multithread lock.
`IVI_SHARED_COMPONENT_ERROR_BASE + 0x1A0`	`IVI_ERROR_CANNOT_CREATE_THREAD_LOCAL`	Could not create thread local.