

**Lx \ Ls Series  
AC Power Source  
IVI Instrument Driver Manual  
LabView, LabWindows/CVI**

## Disclaimer

### DISCLAIMER OF WARRANTY

=====

THIS INSTRUMENT DRIVER, DOCUMENTATION AND ANY ACCOMPANYING FILES ARE PROVIDED "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OR MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. Because of the various hardware and software environments into which this instrument driver may be put, NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE IS OFFERED. The user must assume the entire risk of using the program.

### LIMITATION OF LIABILITY

=====

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL CALIFORNIA INSTRUMENTS CORP. OR ITS REPRESENTATIVES BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF CALIFORNIA INSTRUMENTS CORP. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, CALIFORNIA INSTRUMENTS CORP. ASSUMES NO LIABILITY UNDER ANY PROVISION OF THIS AGREEMENT FOR ANY DAMAGES. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

## Table of Contents

Disclaimer .....	2
Introduction: .....	5
<b>Assumptions:</b> .....	<b>5</b>
<b>Error and Status Information:</b> .....	<b>5</b>
<b>How To Use This Document:</b> .....	<b>6</b>
Function Tree Layout: .....	6
<i>ciLxs_Abort</i> .....	10
<i>ciLxs_CheckAttributeViBoolean</i> .....	12
<i>ciLxs_CheckAttributeViInt32</i> .....	15
<i>ciLxs_CheckAttributeViReal64</i> .....	18
<i>ciLxs_CheckAttributeViSession</i> .....	21
<i>ciLxs_CheckAttributeViString</i> .....	24
<i>ciLxs_ClearAllArbWaveforms</i> .....	27
<i>ciLxs_ClearArbWaveforms</i> .....	29
<i>ciLxs_ClearError</i> .....	31
<i>ciLxs_ClearInterchangeWarnings</i> .....	33
<i>ciLxs_close</i> .....	35
<i>ciLxs_ConfigureAcqTriggerSource</i> .....	37
<i>ciLxs_ConfigureAcquisitionStartTime</i> .....	39
<i>ciLxs_ConfigureClippingLevel</i> .....	41
<i>ciLxs_ConfigureCurrentLimit</i> .....	43
<i>ciLxs_ConfigureFrequencyValue</i> .....	46
<i>ciLxs_ConfigureOutput</i> .....	48
<i>ciLxs_ConfigureOutputALCState</i> .....	50
<i>ciLxs_ConfigureOutputEnabled</i> .....	52
<i>ciLxs_ConfigureOutputPhaseMode</i> .....	54
<i>ciLxs_ConfigureOutputRange</i> .....	56
<i>ciLxs_ConfigureOVP</i> .....	58
<i>ciLxs_ConfigurePhaseAngle</i> .....	61
<i>ciLxs_ConfigurePulse</i> .....	64
<i>ciLxs_ConfigureSlewFrequency</i> .....	66
<i>ciLxs_ConfigureSlewVoltageLevel</i> .....	68
<i>ciLxs_ConfigureSynchronizationPhase</i> .....	70
<i>ciLxs_ConfigureSynchronizationSource</i> .....	72
<i>ciLxs_ConfigureTriggerAngleList</i> .....	74
<i>ciLxs_ConfigureTriggerDelay</i> .....	76
<i>ciLxs_ConfigureTriggeredAngle</i> .....	78
<i>ciLxs_ConfigureTriggeredAngleMode</i> .....	80
<i>ciLxs_ConfigureTriggeredFrequency</i> .....	82
<i>ciLxs_ConfigureTriggeredFrequencyMode</i> .....	84
<i>ciLxs_ConfigureTriggeredFunction</i> .....	86
<i>ciLxs_ConfigureTriggeredFunctionMode</i> .....	88
<i>ciLxs_ConfigureTriggeredSlewFrequency</i> .....	90
<i>ciLxs_ConfigureTriggeredSlewVoltage</i> .....	92
<i>ciLxs_ConfigureTriggeredVoltageLevel</i> .....	94
<i>ciLxs_ConfigureTriggeredVoltageMode</i> .....	97
<i>ciLxs_ConfigureTriggerFrequencyList</i> .....	100
<i>ciLxs_ConfigureTriggerFunctionList</i> .....	102
<i>ciLxs_ConfigureTriggerList</i> .....	104
<i>ciLxs_ConfigureTriggerSource</i> .....	107
<i>ciLxs_ConfigureTriggerTTLOutputList</i> .....	109
<i>ciLxs_ConfigureTriggerVoltageList</i> .....	111
<i>ciLxs_ConfigureTrigSlewFrequencyList</i> .....	114

<i>ciLxs_ConfigureTrigSlewFrequencyMode</i> .....	116
<i>ciLxs_ConfigureTrigSlewVoltageList</i> .....	118
<i>ciLxs_ConfigureTrigSlewVoltageMode</i> .....	121
<i>ciLxs_ConfigureTTLTriggerOutput</i> .....	124
<i>ciLxs_ConfigureVoltageLevel</i> .....	126
<i>ciLxs_ConfigureWaveform</i> .....	129
<i>ciLxs_Disable</i> .....	131
<i>ciLxs_error_message</i> .....	133
<i>ciLxs_error_query</i> .....	139
<i>ciLxs_Fetch</i> .....	141
<i>ciLxs_FetchArray</i> .....	144
<i>ciLxs_FetchHarmonic</i> .....	147
<i>ciLxs_GetAttributeViBoolean</i> .....	150
<i>ciLxs_GetAttributeViInt32</i> .....	153
<i>ciLxs_GetAttributeViReal64</i> .....	156
<i>ciLxs_GetAttributeViSession</i> .....	159
<i>ciLxs_GetAttributeViString</i> .....	162
<i>ciLxs_GetError</i> .....	166
<i>ciLxs_GetNextCoercionRecord</i> .....	169
<i>ciLxs_GetNextInterchangeWarning</i> .....	172
<i>ciLxs_GetPhaseName</i> .....	175
<i>ciLxs_init</i> .....	178
<i>ciLxs_InitiateAcquisition</i> .....	182
<i>ciLxs_InitiateTransient</i> .....	184
<i>ciLxs_InitWithOptions</i> .....	186
<i>ciLxs_InvalidAllAttributes</i> .....	191
<i>ciLxs_LockSession</i> .....	193
<i>ciLxs_Measure</i> .....	196
<i>ciLxs_MeasureArray</i> .....	199
<i>ciLxs_MeasureHarmonic</i> .....	202
<i>ciLxs_QueryArbWaveformCapabilities</i> .....	205
<i>ciLxs_QueryDefinedWaveforms</i> .....	208
<i>ciLxs_QueryMaxCurrentLimit</i> .....	210
<i>ciLxs_QueryMaxVoltageLevel</i> .....	212
<i>ciLxs_QueryOutputState</i> .....	214
<i>ciLxs_QueryTrnsListStatus</i> .....	217
<i>ciLxs_ReadInstrData</i> .....	219
<i>ciLxs_reset</i> .....	221
<i>ciLxs_ResetInterchangeCheck</i> .....	223
<i>ciLxs_ResetOutputProtection</i> .....	225
<i>ciLxs_ResetWithDefaults</i> .....	227
<i>ciLxs_revision_query</i> .....	229
<i>ciLxs_self_test</i> .....	231
<i>ciLxs_SendSoftwareTrigger</i> .....	233
<i>ciLxs_SetAttributeViBoolean</i> .....	235
<i>ciLxs_SetAttributeViInt32</i> .....	239
<i>ciLxs_SetAttributeViReal64</i> .....	243
<i>ciLxs_SetAttributeViSession</i> .....	247
<i>ciLxs_SetAttributeViString</i> .....	251
<i>ciLxs_StoreRecallRegister</i> .....	255
<i>ciLxs_UnlockSession</i> .....	257
<i>ciLxs_WriteArbWaveform</i> .....	260
<i>ciLxs_WriteInstrData</i> .....	262

---

## California Instruments Lx/Ls Series AC Source

---

### Introduction:

This instrument driver provides programming support for California Instr Lx/Ls Series AC Source.

It contains functions for opening, configuring, taking measurements from, and closing the instrument.

---

### Assumptions:

To successfully use this module, the following conditions must be met:

For GPIB instrument drivers:

- the instrument is connected to the GPIB.
- the GPIB address supplied to the initialize function must match the GPIB address of the instrument.

For VXI instrument drivers:

- the instrument is installed in the VXI mainframe and you are using one of the following controller options:
  - Embedded controller
  - MXI
  - MXI2
  - GPIB-VXI
- the logical address supplied to the initialize function must match the logical address of the instrument.

For RS-232 instrument drivers:

- the instrument is connected to the RS-232 interface.
  - the COM port, baud rate, parity, and timeout supplied to the initialize function must match the settings of the instrument.
- 

### Error and Status Information:

Each function in this instrument driver returns a status code that either indicates success or describes an error or warning condition.

Your program should examine the status code from each call to an instrument driver function to determine if an error occurred.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

The description of each instrument driver function lists possible error codes and their meanings

---

---

## How To Use This Document:

Use this document as a programming reference manual.  
It describes each function in the

California Instr Lx/Ls Series AC Source

instrument. The functions appear in alphabetical order, with a description of the function and its C syntax, a description of each parameter, and a list of possible error codes.

---

## Function Tree Layout:

Class/Panel Name:	Function Name:
Initialize	ciLxs_init
Initialize With Options	ciLxs_InitWithOptions
Configuration	
Output	
Configure Output	ciLxs_ConfigureOutput
Configure Output Enabled	ciLxs_ConfigureOutputEnabled
Configure Output ALC State	ciLxs_ConfigureOutputALCState
Configure Output Range	ciLxs_ConfigureOutputRange
Configure Current Limit	ciLxs_ConfigureCurrentLimit
Configure OVP	ciLxs_ConfigureOVP
Configure Voltage Level	ciLxs_ConfigureVoltageLevel
Configure Frequency Value	ciLxs_ConfigureFrequencyValue
Configure Output Phase Mode	ciLxs_ConfigureOutputPhaseMode
Configure Phase Angle	ciLxs_ConfigurePhaseAngle
Configure Waveform	ciLxs_ConfigureWaveform
Configure Clipping Level	ciLxs_ConfigureClippingLevel
Configure Slew Voltage Level	
ciLxs_ConfigureSlewVoltageLevel	
Configure Slew Frequency	ciLxs_ConfigureSlewFrequency
Get Phase Name	ciLxs_GetPhaseName
Arbitrary Waveform	
Write Arb Waveform	ciLxs_WriteArbWaveform
Query Arb Waveform Capabilities	
ciLxs_QueryArbWaveformCapabilities	
Clear All Arb Waveforms	ciLxs_ClearAllArbWaveforms
Clear Arb Waveforms	ciLxs_ClearArbWaveforms
Query Defined Waveforms	ciLxs_QueryDefinedWaveforms
Acquisition	
Configure Acq Trigger Source	
ciLxs_ConfigureAcqTriggerSource	
Configure Acq Start Time	
ciLxs_ConfigureAcquisitionStartTime	
Triggering	
Configure Trigger Source	ciLxs_ConfigureTriggerSource
Configure Trigger Delay	ciLxs_ConfigureTriggerDelay

```

        Configure TTL Trigger Output
ciLxs_ConfigureTTLTriggerOutput
        Configure Triggered Volt
ciLxs_ConfigureTriggeredVoltageLevel
        Configure Triggered Freq
ciLxs_ConfigureTriggeredFrequency
        Configure Triggered Func
ciLxs_ConfigureTriggeredFunction
        Configure Triggered Angle
ciLxs_ConfigureTriggeredAngle
        Configure Triggered Slew Volt
ciLxs_ConfigureTriggeredSlewVoltage
        Configure Triggered Slew Freq
ciLxs_ConfigureTriggeredSlewFrequency
        Mode
        Configure Triggered Volt Mode
ciLxs_ConfigureTriggeredVoltageMode
        Configure Triggered Freq Mode
ciLxs_ConfigureTriggeredFrequencyMode
        Configure Triggered Func Mode
ciLxs_ConfigureTriggeredFunctionMode
        Configure Triggered Angle Mode
ciLxs_ConfigureTriggeredAngleMode
        Configure Trig Slew Volt Mode
ciLxs_ConfigureTrigSlewVoltageMode
        Configure Trig Slew Freq Mode
ciLxs_ConfigureTrigSlewFrequencyMode
        Pulse Trigger
        Configure Pulse
ciLxs_ConfigurePulse
        List Trigger
        Configure Trigger List
ciLxs_ConfigureTriggerList
        Configure Trigger Volt List
ciLxs_ConfigureTriggerVoltageList
        Configure Trigger Freq List
ciLxs_ConfigureTriggerFrequencyList
        Configure Trigger Func List
ciLxs_ConfigureTriggerFunctionList
        Configure Trigger Angle List
ciLxs_ConfigureTriggerAngleList
        Configure Trig Slew Volt List
ciLxs_ConfigureTrigSlewVoltageList
        Configure Trig Slew Freq List
ciLxs_ConfigureTrigSlewFrequencyList
        Configure Trig TTL Output List
ciLxs_ConfigureTriggerTTLOutputList
        Synchronization
        Configure Synch Source
ciLxs_ConfigureSynchronizationSource
        Configure Synch Phase
ciLxs_ConfigureSynchronizationPhase
        Set/Get/Check Attribute
        Set Attribute
        Set Attribute ViInt32
ciLxs_SetAttributeViInt32
        Set Attribute ViReal64
ciLxs_SetAttributeViReal64
        Set Attribute ViString
ciLxs_SetAttributeViString
        Set Attribute ViBoolean
ciLxs_SetAttributeViBoolean
        Set Attribute ViSession
ciLxs_SetAttributeViSession
        Get Attribute

```

Get Attribute ViInt32	ciLxs_GetAttributeViInt32
Get Attribute ViReal64	ciLxs_GetAttributeViReal64
Get Attribute ViString	ciLxs_GetAttributeViString
Get Attribute ViBoolean	ciLxs_GetAttributeViBoolean
Get Attribute ViSession	ciLxs_GetAttributeViSession
Check Attribute	
Check Attribute ViInt32	ciLxs_CheckAttributeViInt32
Check Attribute ViReal64	ciLxs_CheckAttributeViReal64
Check Attribute ViString	ciLxs_CheckAttributeViString
Check Attribute ViBoolean	ciLxs_CheckAttributeViBoolean
Check Attribute ViSession	ciLxs_CheckAttributeViSession
Action/Status	
Send Software Trigger	ciLxs_SendSoftwareTrigger
Query Max Current Limit	ciLxs_QueryMaxCurrentLimit
Query Max Voltage Level	ciLxs_QueryMaxVoltageLevel
Query Transient List Status	ciLxs_QueryTrnsListStatus
Query Output State	ciLxs_QueryOutputState
Reset Output Protection	ciLxs_ResetOutputProtection
Store Recall Register	ciLxs_StoreRecallRegister
Measure	
Measure	ciLxs_Measure
Fetch	ciLxs_Fetch
Harmonic Measurement	
Measure Harmonic	ciLxs_MeasureHarmonic
Fetch Harmonic	ciLxs_FetchHarmonic
Array Measurement	
Measure Array	ciLxs_MeasureArray
Fetch Array	ciLxs_FetchArray
Low-Level	
Initiate Transient	ciLxs_InitiateTransient
Initiate Acquisition	ciLxs_InitiateAcquisition
Abort	ciLxs_Abort
Utility	
Self-Test	ciLxs_self_test
Reset	ciLxs_reset
Reset With Defaults	ciLxs_ResetWithDefaults
Disable	ciLxs_Disable
Revision Query	ciLxs_revision_query
Error-Query	ciLxs_error_query
Error Message	ciLxs_error_message
Invalidate All Attributes	ciLxs_InvalidateAllAttributes
Error	
Get Error	ciLxs_GetError
Clear Error	ciLxs_ClearError
Coercion Info	
Get Next Coercion Record	ciLxs_GetNextCoercionRecord
Interchangeability Info	
Get Next Interchange Warning	ciLxs_GetNextInterchangeWarning
Clear Interchange Warnings	ciLxs_ClearInterchangeWarnings
Reset Interchange Check	ciLxs_ResetInterchangeCheck
Locking	
Lock Session	ciLxs_LockSession
Unlock Session	ciLxs_UnlockSession
Instrument I/O	
Write Instrument Data	ciLxs_WriteInstrData

---

Read Instrument Data	ciLxs_ReadInstrData
Close	ciLxs_close

---

#### California Instr Lx/Ls Series AC Source

This instrument driver provides programming support for the California Instrument iL series of AC Power Supplies. The driver contains all the functions that IVI and VXIplug&play require. In addition, the driver contains high-level functions that configure the power supply and generate output in a single operation. The driver also contains lower level functions that configure the power supply and initiate the output changes in separate operations.

Note: This driver requires the VISA and IVI libraries.

---

The following functions are in alphabetical order.

---

***ciLxs\_Abort***

```
ViStatus ciLxs_Abort (ViSession instrumentHandle);
```

## Purpose

This function aborts all pending output changes.

## Parameter List

```
instrumentHandle
```

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

---

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_CheckAttributeViBoolean***

```
ViStatus ciLxs_CheckAttributeViBoolean (ViSession instrumentHandle,
                                       ViChar _VI_FAR channelName[],
                                       ViAttr attributeID,
                                       ViBoolean attributeValue);
```

## Purpose

This function checks the validity of a value you specify for a ViBoolean attribute.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## channelName

Variable Type            ViChar[]

If the attribute is channel-based, this parameter specifies the name of the channel on which to check the attribute value. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViBoolean type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViBoolean

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You

examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_CheckAttributeViInt32***

```
ViStatus ciLxs_CheckAttributeViInt32 (ViSession instrumentHandle,
                                       ViChar _VI_FAR channelName[],
                                       ViAttr attributeID,
                                       ViInt32 attributeValue);
```

## Purpose

This function checks the validity of a value you specify for a ViInt32 attribute.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## channelName

Variable Type            ViChar[]

If the attribute is channel-based, this parameter specifies the name of the channel on which to check the attribute value. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViInt32

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either

indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_CheckAttributeViReal64***

```
ViStatus ciLxs_CheckAttributeViReal64 (ViSession instrumentHandle,
                                       ViChar _VI_FAR channelName[],
                                       ViAttr attributeID,
                                       ViReal64 attributeValue);
```

## Purpose

This function checks the validity of a value you specify for a ViReal64 attribute.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## channelName

Variable Type            ViChar[]

If the attribute is channel-based, this parameter specifies the name of the channel on which to check the attribute value. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViReal64 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViReal64

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You

examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_CheckAttributeViSession***

```
ViStatus ciLxs_CheckAttributeViSession (ViSession instrumentHandle,
                                       ViChar _VI_FAR channelName[],
                                       ViAttr attributeID,
                                       ViSession attributeValue);
```

## Purpose

This function checks the validity of a value you specify for a ViSession attribute.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## channelName

Variable Type            ViChar[]

If the attribute is channel-based, this parameter specifies the name of the channel on which to check the attribute value. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## attributeID

Variable Type      ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViSession type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type      ViSession

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either

indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_CheckAttributeViString***

```
ViStatus ciLxs_CheckAttributeViString (ViSession instrumentHandle,
                                       ViChar _VI_FAR channelName[],
                                       ViAttr attributeID,
                                       ViChar _VI_FAR attributeValue[]);
```

**Purpose**

This function checks the validity of a value you specify for a ViString attribute.

**Parameter List****instrumentHandle**

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

**channelName**

Variable Type	ViChar[]
---------------	----------

If the attribute is channel-based, this parameter specifies the name of the channel on which to check the attribute value. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

**Notes:**

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

**attributeID**

Variable Type      ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViString type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type      ViChar[]

Pass the value which you want to verify as a valid value for the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You

examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
-----	
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ClearAllArbWaveforms***

```
ViStatus ciLxs_ClearAllArbWaveforms (ViSession instrumentHandle);
```

## Purpose

This function deletes all user-defined waveforms.

## Parameter List

instrumentHandle

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
------------------------	-------------------

---

3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIIPnP	Driver Errors

---

***ciLxs\_ClearArbWaveforms***

```
ViStatus ciLxs_ClearArbWaveforms (ViSession instrumentHandle,
                                  ViString waveformName);
```

## Purpose

This function deletes individual user-defined waveforms.

## Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

waveformName

Variable Type            ViString

This control specifies the name of the waveform to be defined.

Valid Range:

String with less than 15 characters.

Default Value: "ArbWave"

Notes:

The Waveform Name will be truncated if it exceeds 15 characters.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success

Positive Values            Warnings  
 Negative Values         Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:

BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI        Warnings
3FFF0000 to 3FFFFFFF	VISA      Warnings
3FFC0000 to 3FFCFFFF	VXIPnP   Driver Warnings
BFFA0000 to BFFA1FFF	IVI        Errors
BFFF0000 to BFFFFFFF	VISA      Errors
BFFC0000 to BFFCFFFF	VXIPnP   Driver Errors

***ciLxs\_ClearError***

```
ViStatus ciLxs_ClearError (ViSession instrumentHandle);
```

## Purpose

This function clears the error code and error description for the IVI session. If the user specifies a valid IVI session for the instrument handle 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 VI\_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 ciLxs\_GetError function, which occurs when a call to ciLxs\_init or ciLxs\_InitWithOptions fails.

## Parameter List

instrumentHandle

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
-----	
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
--------	-------------

---

## WARNINGS:

None

## ERRORS:

BFFA4001 Histogram is not enabled.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviScope Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviScope Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_ClearInterchangeWarnings***

```
ViStatus ciLxs_ClearInterchangeWarnings (ViSession instrumentHandle);
```

## Purpose

This function clears the list of current interchange warnings.

## Parameter List

```
instrumentHandle
```

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetErrorInfo` function. To clear the error information from the driver, call the `ciLxs_ClearErrorInfo` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
WARNINGS:	
none	
ERRORS:	
none	

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the

particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviDCPwr Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviDCPwr Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_close***

```
ViStatus ciLxs_close (ViSession instrumentHandle);
```

## Purpose

This function performs the following operations:

- Closes the instrument I/O session.
- Destroys the instrument driver session and all of its attributes.
- Deallocates any memory resources the driver uses.

## Notes:

- (1) You must unlock the session before calling `ciLxs_close`.
- (2) After calling `ciLxs_close`, you cannot use the instrument driver again until you call `ciLxs_init` or `ciLxs_InitWithOptions`.

## Parameter List

`instrumentHandle`

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
-----	
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

## ***ciLxs\_ConfigureAcqTriggerSource***

```
ViStatus ciLxs_ConfigureAcqTriggerSource (ViSession instrumentHandle,  
                                          ViInt32 source);
```

### Purpose

This function configures the trigger source for a triggered measurement sequence.

### Parameter List

#### instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

#### source

Variable Type            ViInt32

Pass the trigger source for a triggered measurement sequence. The driver sets the `CILXS_ATTR_ACQUISITION_TRIGGER_SOURCE` attribute to this value.

Defined Values:

`CILXS_VAL_SOFTWARE_TRIG` - The power supply waits until you call the `ciLxs_SendSoftwareTrigger` function.

`CILXS_VAL_TRIG_EXTERNAL` - The power supply waits for a trigger on the external trigger input.

`CILXS_VAL_TRIG_TTLT` - The power supply waits for a signal driving the Trigger Out BNC

Default Value: `CILXS_VAL_SOFTWARE_TRIG`

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the

ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureAcquisitionStartTime***

```
ViStatus ciLxs_ConfigureAcquisitionStartTime (ViSession instrumentHandle,
                                             ViReal64 startTime);
```

## Purpose

This function configures the length of time from the acquire trigger event to the first point in the record.

## Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

startTime

Variable Type            ViReal64

Specifies the length of time from the acquire trigger event to the first point in the record. If the value is positive, the first point in the record occurs after the acquire trigger event. If the value is negative, the first point in the record occurs before the acquire trigger event. The driver sets the `CILXS_ATTR_ACQUISITION_START_TIME` to this value.

Units: seconds

Valid Range:

Range depends on the `CILXS_ATTR_ACQUISITION_TIME_INTERVAL` attribute. Following formula is valid.

```
min = -4095 * CILXS_ATTR_ACQUISITION_TIME_INTERVAL
max = 2e9 * CILXS_ATTR_ACQUISITION_TIME_INTERVAL
```

Default Value: 0.0

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

## ***ciLxs\_ConfigureClippingLevel***

```
ViStatus ciLxs_ConfigureClippingLevel (ViSession instrumentHandle,  
                                       ViReal64 clippingLevel);
```

### Purpose

This function configures the clipping level when a clipped sine output waveform is selected.

### Parameter List

instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

clippingLevel

Variable Type          ViReal64

Pass the clipping level when a clipped sine output waveform is selected. The driver sets the `CILXS_ATTR_CLIPPING_LEVEL` attribute to this value.

Units: percentage

Valid THD Range: 0.0% to 43.0%(1.5 to 100.0 clip level)

Default Value: 0.0%

Note:

(1) This parameter is ignored when function `CILXS_VAL_CLIPPED_SINE ("CSINUSOID")` is not selected.

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureCurrentLimit***

```
ViStatus ciLxs_ConfigureCurrentLimit (ViSession instrumentHandle,
                                     ViChar _VI_FAR phase[],
                                     ViInt32 behavior, ViReal64 limit);
```

## Purpose

This function configures the current limit. You specify the output current limit value and the behavior of the power supply when the output current is greater than or equal to that value.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type      ViChar[]

Pass the virtual phase name that you assign to the instrument in the Configuration Utility.

Virtual phase names are aliases for instrument-specific phase strings. The instrument-specific phase strings can differ from one instrument to another. Virtual phase names allow you to use and swap instruments without having to change the phase names in your source code. You assign a virtual phase name to an instrument-specific phase through the Configuration Utility. This control accepts virtual phase names you have assigned to the specific instrument you are using. It also accepts the instrument-specific phase names.

Default Value: ""

## Notes:

(1) You can specify the phase name as a string variable or as a literal enclosed in double quotes.

## behavior

Variable Type      ViInt32

Pass the behavior you want the power supply to exhibit when the output current is greater than or equal to the value of the limit parameter on the specified phase. The driver uses this value to set

the CILXS\_ATTR\_CURRENT\_LIMIT\_BEHAVIOR attribute.

Defined Values:

CILXS\_VAL\_CURRENT\_REGULATE - Regulatory limit  
 CILXS\_VAL\_CURRENT\_TRIP - Trip limit

Default Value: CILXS\_VAL\_CURRENT\_REGULATE

limit

Variable Type ViReal64

Pass the rms current limit of the specified output phase. The driver uses this value to set the CILXS\_ATTR\_CURRENT\_LIMIT attribute.

Units: amps

Valid Range (1 phase mode): 0.0 to 20.0 (3000iL)  
 0.0 to 30.0 (4500iL)  
 0.0 to 32.0 (4801iL)

Valid Range (3 phase mode): 0.0 to 6.7 (3000iL)  
 0.0 to 10.0 (4500iL)

Default Value: 0.0

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_ConfigureFrequencyValue***

```
ViStatus ciLxs_ConfigureFrequencyValue (ViSession instrumentHandle,
                                       ViReal64 value);
```

## Purpose

This function configures the AC RMS voltage level that the power supply attempts to generate.

## Parameter List

instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

value

Variable Type          ViReal64

Pass the AC frequency you want the AC source to attempt to generate. The driver sets the `CILXS_ATTR_FREQUENCY` attribute to this value.

Units: Hz

Valid Range:  
45.0 to 5000.0

Default Value: 50.0 Hz

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
-----	

```

0                Success
Positive Values  Warnings
Negative Values  Errors
    
```

This driver defines the following status codes:

```

Status    Description
-----
ERRORS:
BFFA1001  The trigger source is not software trigger.
    
```

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

## ***ciLxs\_ConfigureOutput***

```
ViStatus ciLxs_ConfigureOutput (ViSession instrumentHandle,  
                               ViInt32 numberOfPhases);
```

### Purpose

Configures the number of outputs phases for the AC source.

### Note:

(1) Execution of this function disables all outputs, clears lists and \*RCL states to the initialization default values, reconfigures current readback and programming calibration constants, and reboots the product.

(2) The AC source must be calibrated in the three phase mode to properly execute this function.

### Parameter List

#### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

#### numberOfPhases

Variable Type      ViInt32

Specifies the number of output phases for ac sources that have single-phase and three-phase switchable capability. The driver uses this value to set the CILXS\_ATTR\_OUTPUT\_PHASE\_COUNT attribute.

#### Valid Values:

CILXS\_VAL\_1\_PHASE - Single phase mode  
CILXS\_VAL\_3\_PHASE - Three phase mode

Default Value: CILXS\_VAL\_1\_PHASE

#### Note:

1) When number of phase is changed, driver waits 15 seconds. Instrument requires this time for changing output phase.

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureOutputALCState***

```
ViStatus ciLxs_ConfigureOutputALCState (ViSession instrumentHandle,
                                       ViInt32 ALCState);
```

## Purpose

This function enables or disables the AC source output. The state of a disabled output is an output voltage amplitude set to 0 volts, with output relays opened.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## ALCState

Variable Type            ViInt32

Pass whether you want to enable the ALC on the AC source. The driver uses this value to set the `CILXS_ATTR_OUTPUT_ALC_STATE` attribute.

## Valid Values:

<code>CILXS_ALC_OFF</code>	(0) - Disable ALC on the output
<code>CILXS_ALC_ON</code>	(1) - Enable ALC on the output(output will fault if output voltage is far apart from programmed value)
<code>CILXS_ALC_REGULATE</code>	(2) - Enable ALC on the output(output will NOT fault if output voltage is far apart from programmed value)

Default Value: `CILXS_ALC_REGULATE`

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureOutputEnabled***

```
ViStatus ciLxs_ConfigureOutputEnabled (ViSession instrumentHandle,
                                       ViBoolean enabled);
```

## Purpose

This function enables or disables the AC source output. The state of a disabled output is an output voltage amplitude set to 0 volts, with output relays opened.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## enabled

Variable Type            ViBoolean

Pass whether you want the signal the power supply produces on a output. The driver uses this value to set the `CILXS_ATTR_OUTPUT_ENABLED` attribute.

## Valid Values:

- VI\_TRUE (1) - Enable the output
- VI\_FALSE (0) - Disable the output

Default Value: VI\_TRUE

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings

Negative Values                      Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:

BFFA1001    The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings
3FFF0000 to 3FFFFFFF	VISA          Warnings
3FFC0000 to 3FFCFFFF	VXIPnP      Driver Warnings
-----	
BFFA0000 to BFFA1FFF	IVI            Errors
BFFF0000 to BFFFFFFF	VISA          Errors
BFFC0000 to BFFCFFFF	VXIPnP      Driver Errors

***ciLxs\_ConfigureOutputPhaseMode***

```
ViStatus ciLxs_ConfigureOutputPhaseMode (ViSession instrumentHandle,
                                         ViInt32 phaseMode);
```

## Purpose

This function selects the output phase mode for three phase systems. Available options are single or three phase mode. In three phase mode, the phase coupling can be set to COUPLED or UNCOUPLED.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phaseMode

Variable Type            ViInt32

Specifies the number of output phases for ac sources that have single-phase and three-phase switchable capability. The driver uses this value to set the `CILXS_ATTR_OUTPUT_PHASE_COUNT` attribute.

## Valid Values:

`CILXS_VAL_1_PHASE` - Single phase mode  
`CILXS_VAL_3_PHASE` - Three phase mode

Default Value: `CILXS_VAL_1_PHASE`

## Note:

1) When number of phase is changed, driver waits 15 seconds. Instrument requires this time for changing output phase.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

### ***ciLxs\_ConfigureOutputRange***

```
ViStatus ciLxs_ConfigureOutputRange (ViSession instrumentHandle,  
                                     ViReal64 voltageRange);
```

#### Purpose

Configures the power supply's output range. You specify whether you want to configure the voltage or current range, and the value to which to set the range.

#### Notes:

- 1) Setting a voltage range can invalidate a previously configured current range.
- 2) Setting a current range can invalidate a previously configured voltage range.

#### Parameter List

##### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

##### voltageRange

Variable Type      ViReal64

Pass the range in which you want the AC source to operate.

Units: volts (for voltage range)

Valid Voltage (rms) Range: 0.0 to 400.0

Default Value: 135.0

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError`

function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureOVP***

```
ViStatus ciLxs_ConfigureOVP (ViSession instrumentHandle,
                             ViChar _VI_FAR phase[], ViBoolean enabled,
                             ViReal64 limit);
```

## Purpose

This function configures the power supply's over-voltage protection. You specify the over-voltage limit and the behavior of the power supply when the output voltage is greater than or equal to that value.

When the enabled parameter is VI\_FALSE, the limit parameter does not affect the instrument's behavior, and the driver ignores the limit parameter.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type      ViChar[]

Pass the virtual phase name that you assign to the instrument in the Configuration Utility.

Virtual phase names are aliases for instrument-specific phase strings. The instrument-specific phase strings can differ from one instrument to another. Virtual phase names allow you to use and swap instruments without having to change the phase names in your source code. You assign a virtual phase name to an instrument-specific phase through the Configuration Utility. This control accepts virtual phase names you have assigned to the specific instrument you are using. It also accepts the instrument-specific phase names.

Default Value: ""

## Notes:

(1) You can specify the phase name as a string variable or as a literal enclosed in double quotes.

## enabled

Variable Type      ViBoolean

Pass whether you want to enable or disable the OVP limit. The driver uses this value to set the CILXS\_ATTR\_OVP\_ENABLED attribute.

Valid Values:

- VI\_TRUE (1) - Enable OVP limit
- VI\_FALSE (0) - Disable OVP limit

Default Value: VI\_TRUE

limit

Variable Type            ViReal64

Pass the over-voltage protection limit you want to use for the specified phase. The driver uses this value to set the CILXS\_ATTR\_OVP\_LIMIT attribute.

Units: volts

Valid Range:  
0.0 to 500.0

Default Value: 500.0 volts

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional

status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_ConfigurePhaseAngle***

```
ViStatus ciLxs_ConfigurePhaseAngle (ViSession instrumentHandle,
                                     ViChar _VI_FAR phase[],
                                     ViReal64 angle);
```

## Purpose

This function configures the phase of the output voltage waveform relative to an internal reference.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type            ViChar[]

Pass the virtual phase name that you assign to the instrument in the Configuration Utility.

Virtual phase names are aliases for instrument-specific phase

strings. The instrument-specific phase strings can differ from one instrument to another. Virtual phase names allow you to use and swap instruments without having to change the phase names in your source code. You assign a virtual phase name to an instrument-specific phase through the Configuration Utility. This control accepts virtual phase names you have assigned to the specific instrument you are using. It also accepts the instrument-specific phase names.

Default Value: ""

## Notes:

(1) You can specify the phase name as a string variable or as a literal enclosed in double quotes.

## angle

Variable Type            ViReal64

Pass the phase of the output voltage waveform relative to an internal reference. Positive phase angles are used to program the leading phase, and negative phase angles are used to program the lagging

phase. The phase angle is set in degrees. The driver uses this value to set the CILXS\_ATTR\_PHASE\_ANGLE attribute.

Units: degrees

Valid Range:  
-360.0 to 360.0

Default Value: 0.0 degree

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors



### ***ciLxs\_ConfigurePulse***

```
ViStatus ciLxs_ConfigurePulse (ViSession instrumentHandle,  
                               ViInt32 count, ViReal64 width,  
                               ViReal64 period);
```

#### Purpose

This function configures the generation of output pulses.

#### Parameter List

##### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

##### count

Variable Type      ViInt32

Pass the number of pulses that are output when a triggered output transient occurs. The driver sets the `CILXS_ATTR_TRIGGER_PULSE_COUNT` attribute to this value.

Valid Range: 1 to 2147483647

Default Value: 1

##### width

Variable Type      ViReal64

Pass the the width of a transient output pulse. The driver sets the `CILXS_ATTR_TRIGGER_PULSE_WIDTH` attribute to this value.

Units: Seconds

Valid Range:

3-phase models: 0 to 1.07533e6

1-phase models: 0 to 4.30133e5

Default Value: 0.01667

##### period

Variable Type      ViReal64

Pass the period of a triggered output transient. The driver sets the

CILXS\_ATTR\_TRIGGER\_PULSE\_PERIOD attribute to this value.

Units: Seconds

Valid Range:

- 3-phase models: 0 to 1.07533E6
- 1-phase models: 0 to 4.30133E5

Default Value: 0.03333

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

### ***ciLxs\_ConfigureSlewFrequency***

```
ViStatus ciLxs_ConfigureSlewFrequency (ViSession instrumentHandle,  
                                       ViReal64 slewFrequencyRate);
```

#### Purpose

This function configures the rate at which frequency changes for all programmed changes in output frequency.

#### Parameter List

instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

slewFrequencyRate

Variable Type          ViReal64

Pass the rate at which frequency changes for all programmed changes in output frequency. The driver sets the `CILXS_ATTR_SLEW_FREQUENCY_RATE` attribute to this value.

Units: hertz per seconds

Valid Range: 0.0 to 9.9e37

Default Value: -1.0

#### NOTE:

If passed value is lower than zero then instrument will set maximum possible range (INFINITY).

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureSlewVoltageLevel***

```
ViStatus ciLxs_ConfigureSlewVoltageLevel (ViSession instrumentHandle,
                                          ViChar _VI_FAR phase[],
                                          ViReal64 slewVoltageRate);
```

## Purpose

This function configures the slew rate for all programmed changes in the ac rms output voltage level of the AC source.

## Parameter List

## instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type          ViChar[]

Pass the virtual phase name that you assign to the instrument in the Configuration Utility.

Virtual phase names are aliases for instrument-specific phase strings. The instrument-specific phase strings can differ from one instrument to another. Virtual phase names allow you to use and swap instruments without having to change the phase names in your source code. You assign a virtual phase name to an instrument-specific phase through the Configuration Utility. This control accepts virtual phase names you have assigned to the specific instrument you are using. It also accepts the instrument-specific phase names.

Default Value: ""

## Notes:

(1) You can specify the phase name as a string variable or as a literal enclosed in double quotes.

## slewVoltageRate

Variable Type          ViReal64

Pass the slew rate for all programmed changes in the ac rms output voltage level of the AC source. The driver sets the CILXS\_ATTR\_SLEW\_VOLTAGE\_RATE attribute to this value.

Units: volts per seconds

Valid Range: 0.0 to 9.9e37

Default Value: -1.0

NOTE:

If passed value is lower than zero then instrument will set maximum possible range (INFINITY).

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureSynchronizationPhase***

```
ViStatus ciLxs_ConfigureSynchronizationPhase (ViSession instrumentHandle,
                                             ViReal64 angle);
```

## Purpose

This function configures the phase angle with respect to an internal phase reference at which PHASE:SYNChronous:SOURce becomes true.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## angle

Variable Type            ViReal64

Pass the phase angle with respect to an internal phase reference.

The driver sets the CILXS\_ATTR\_TRIGGER\_SYNCHRONIZATION\_PHASE attribute to this value.

Units: degrees

Valid Range:  
-360.0 to 360.0

Default Value: 0.0

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureSynchronizationSource***

```
ViStatus ciLxs_ConfigureSynchronizationSource
    (ViSession instrumentHandle, ViInt32 source);
```

## Purpose

This function selects the synchronizing trigger source in generating a step, pulse, or list.

## Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

source

Variable Type            ViInt32

Pass the synchronizing trigger source in generating a step, pulse, or list output. The driver sets the `CILXS_ATTR_TRIGGER_SYNCHRONIZATION_SOURCE` attribute to this value.

Defined Values:

`CILXS_VAL_SYNCHRONIZATION_SOURCE_IMMEDIATE` - Starts the transient output immediately, unless a delay time other than 0 has been specified by `ciLxs_ConfigureTriggerDelay` function. In this case the transient output starts after the expiration of the delay time.

`CILXS_VAL_SYNCHRONIZATION_SOURCE_PHASE` - starts the transient output at the reference phase set by `ciLxs_ConfigureSynchronizationPhase` function.

Default Value: `CILXS_VAL_SYNCHRONIZATION_SOURCE_IMMEDIATE`

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError`

function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggerAngleList***

```
ViStatus ciLxs_ConfigureTriggerAngleList (ViSession instrumentHandle,
                                          ViChar _VI_FAR phase[],
                                          ViInt32 listSize,
                                          ViReal64 _VI_FAR angles[]);
```

**Purpose**

This function configures the sequence of phase list points.

**Parameter List****instrumentHandle**

Variable Type          ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

**phase**

Variable Type          ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: "PHASE1"

**Notes:**

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

**listSize**

Variable Type          ViInt32

The number of list points.

**angles**

Variable Type            ViReal64[]

The sequence of angles list points.

The phase points are given in the command parameters, which are separated by commas. The order in which the points are entered determines the sequence in which they are output when a list is triggered.

Units: degrees  
 Valid Range:  
       -360.0 to 360.0

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI        Warnings
3FFF0000 to 3FFFFFFF	VISA      Warnings
3FFC0000 to 3FFCFFFF	VXIPnP    Driver Warnings
BFFA0000 to BFFA1FFF	IVI        Errors
BFFF0000 to BFFFFFFF	VISA      Errors
BFFC0000 to BFFCFFFF	VXIPnP    Driver Errors

## ***ciLxs\_ConfigureTriggerDelay***

```
ViStatus ciLxs_ConfigureTriggerDelay (ViSession instrumentHandle,  
                                     ViReal64 triggerDelay);
```

### Purpose

This function configures the time delay between the detection of a trigger signal and the start of any corresponding trigger action.

### Parameter List

#### instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

#### triggerDelay

Variable Type            ViReal64

Pass the time delay between the detection of a trigger signal and the start of any corresponding trigger action. After the time delay has elapsed, the trigger is implemented unless the trigger system is also waiting for a sync signal that has been specified by `ciLxs_ConfigureSynchronizationPhase`. The driver sets the `CILXS_ATTR_TRIGGER_DELAY` attribute to this value.

Units: seconds

#### Defined Values:

3-phase models: 0 to 1.07533e6

1-phase models: 0 to 4.30133e5

Default Value: 0.0 second

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError`

function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggeredAngle***

```
ViStatus ciLxs_ConfigureTriggeredAngle (ViSession instrumentHandle,
                                       ViChar _VI_FAR phaseName[],
                                       ViReal64 angle);
```

## Purpose

This function configures the output phase when a triggered step or pulse transient occurs. The phase of the output voltage waveform is expressed relative to an internal reference.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## phaseName

Variable Type            ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## angle

Variable Type            ViReal64

Pass the output phase when a triggered step or pulse transient occurs. The phase of the output voltage waveform is expressed relative to an internal reference. The phase angle is programmed in

degrees. Positive phase angles are used to program the leading phase, and negative phase angles are used to program the lagging phase. The driver sets the CILXS\_ATTR\_TRIGGERED\_PHASE\_ANGLE attribute to this value.

Units: degrees

Valid Range:  
-360.0 to 360.0

Default Value: 0.0 degrees

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggeredAngleMode***

```
ViStatus ciLxs_ConfigureTriggeredAngleMode (ViSession instrumentHandle,
                                           ViChar _VI_FAR phaseName[],
                                           ViInt32 transientMode);
```

## Purpose

This function configures the output phase controlled during a triggered output transient.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phaseName

Variable Type            ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## transientMode

Variable Type            ViInt32

Specifies how the output phase angle is controlled during a triggered output transient. The driver sets the `CILXS_ATTR_TRIGGERED_PHASE_ANGLE_MODE` attribute to this value.

Valid Values:

- CILXS\_VAL\_TRIGGER\_MODE\_FIX - The output phase is unaffected by a triggered output transient.
- CILXS\_VAL\_TRIGGER\_MODE\_STEP - The output phase is programmed to the value set by `ciLxs_ConfigureTriggeredAngle` when a triggered transient occurs.
- CILXS\_VAL\_TRIGGER\_MODE\_PULSE - The output phase is changed to the value set by `ciLxs_ConfigureTriggeredAngle` for a duration determined by the `ciLxs_ConfigurePulse` function.
- CILXS\_VAL\_TRIGGER\_MODE\_LIST - The waveform shape is controlled by the phase list when a triggered transient occurs.

Default Value:

CILXS\_VAL\_TRIGGER\_MODE\_FIX

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred. To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggeredFrequency***

```
ViStatus ciLxs_ConfigureTriggeredFrequency (ViSession instrumentHandle,
                                           ViReal64 triggeredFrequency);
```

## Purpose

This function configures the frequency that the output will be set to during a triggered step or pulse transient.

## Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

triggeredFrequency

Variable Type            ViReal64

Pass the frequency that the output will be set to during a triggered step or pulse transient. The driver sets the CILXS\_ATTR\_TRIGGERED\_FREQUENCY attribute to this value.

Units: hertz

Valid Range:  
45.0 to 5000.0

Default Value: 60.0 hertz

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success

Positive Values            Warnings  
 Negative Values         Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	
3FFA0000 to 3FFA1FFF	IVI        Warnings
3FFF0000 to 3FFFFFFF	VISA      Warnings
3FFC0000 to 3FFCFFFF	VXIPnP   Driver Warnings
BFFA0000 to BFFA1FFF	IVI        Errors
BFFF0000 to BFFFFFFF	VISA      Errors
BFFC0000 to BFFCFFFF	VXIPnP   Driver Errors

***ciLxs\_ConfigureTriggeredFrequencyMode***

```
ViStatus ciLxs_ConfigureTriggeredFrequencyMode
    (ViSession instrumentHandle, ViInt32 transientMode);
```

## Purpose

This function configures the output frequency controlled during a triggered output transient.

## Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

transientMode

Variable Type            ViInt32

Specifies how the output frequency is controlled during a triggered output transient. The driver sets the `CILXS_ATTR_TRIGGERED_FREQUENCY_MODE` attribute to this value.

## Valid Values:

- `CILXS_VAL_TRIGGER_MODE_FIX` - The output frequency is unaffected by a triggered output transient.
- `CILXS_VAL_TRIGGER_MODE_STEP` - The output frequency is programmed to the value set by `ciLxs_ConfigureTriggeredFrequency` when a triggered transient occurs.
- `CILXS_VAL_TRIGGER_MODE_PULSE` - The output frequency is changed to the value set by `ciLxs_ConfigureTriggeredFrequency` for a duration determined by the `ciLxs_ConfigurePulse` function.
- `CILXS_VAL_TRIGGER_MODE_LIST` - The output frequency is controlled by the frequency list when a triggered transient occurs.

## Default Value:

`CILXS_VAL_TRIGGER_MODE_FIX`

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError`

function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggeredFunction***

```
ViStatus ciLxs_ConfigureTriggeredFunction (ViSession instrumentHandle,
                                           ViChar _VI_FAR function[]);
```

## Purpose

This function configures the shape of the output voltage waveform.

## Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

function

Variable Type            ViChar[]

Pass the shape of the output voltage waveform when a triggered step or pulse transient occurs. The driver sets the CILXS\_ATTR\_TRIGGERED\_FUNCTION attribute to this value.

Valid Function Names:

"SINUSOID", "SQUARE", "CSINUSOID" and user defined functions

Default Value: "SINUSOID"

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings

Negative Values                      Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:  
 BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings
3FFF0000 to 3FFFFFFF	VISA          Warnings
3FFC0000 to 3FFCFFFF	VXIPnP      Driver Warnings
-----	
BFFA0000 to BFFA1FFF	IVI            Errors
BFFF0000 to BFFFFFFF	VISA          Errors
BFFC0000 to BFFCFFFF	VXIPnP      Driver Errors

***ciLxs\_ConfigureTriggeredFunctionMode***

```
ViStatus ciLxs_ConfigureTriggeredFunctionMode
(ViSession instrumentHandle, ViInt32 transientMode);
```

## Purpose

This function configures the waveform shape controlled during a triggered output transient.

## Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

transientMode

Variable Type            ViInt32

Specifies how the waveform shape is controlled during a triggered output transient. The driver sets the `CILXS_ATTR_TRIGGERED_FUNCTION_MODE` attribute to this value.

## Valid Values:

- `CILXS_VAL_TRIGGER_MODE_FIX` - The waveform shape is unaffected by a triggered output transient.
- `CILXS_VAL_TRIGGER_MODE_STEP` - The waveform shape is programmed to the value set by `ciLxs_ConfigureTriggeredFunction` when a triggered transient occurs.
- `CILXS_VAL_TRIGGER_MODE_PULSE` - The waveform shape is changed to the value set by `ciLxs_ConfigureTriggeredFunction` for a duration determined by the by the `ciLxs_ConfigurePulse` function.
- `CILXS_VAL_TRIGGER_MODE_LIST` - The waveform shape is controlled by the waveform shape list when a triggered transient occurs.

## Default Value:

`CILXS_VAL_TRIGGER_MODE_FIX`

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError`

function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

### ***ciLxs\_ConfigureTriggeredSlewFrequency***

```
ViStatus ciLxs_ConfigureTriggeredSlewFrequency
    (ViSession instrumentHandle,
     ViReal64 triggeredSlewFrequencyRate);
```

#### Purpose

This function configures the rate at which frequency changes for all programmed changes in output frequency.

#### Parameter List

##### instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

##### triggeredSlewFrequencyRate

Variable Type            ViReal64

Pass the rate at which frequency changes during a triggered output transient. Instantaneous frequency changes can be obtained by sending maximum or infinity. The driver sets the `CILXS_ATTR_TRIGGERED_SLEW_FREQUENCY_RATE` attribute to this value.

Units: hertz per second

Valid Range: 0.0 to 9.9e37

Default Value: 9.9e37

#### NOTE:

If passed value is lower than zero then instrument will set maximum possible range (INFINITY).

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the

error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggeredSlewVoltage***

```
ViStatus ciLxs_ConfigureTriggeredSlewVoltage (ViSession instrumentHandle,
                                             ViChar _VI_FAR phase[],
                                             ViReal64 slewRateVoltage);
```

## Purpose

This function configures the slew rate for all programmed changes in the AC rms output voltage level of the AC source.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type      ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## slewRateVoltage

Variable Type      ViReal64

Specifies the slew rate that will be set during a triggered step or pulse transient. A parameter of maximum or infinity will set the slew to its maximum possible rate. The driver sets the `CILXS_ATTR_TRIGGERED_SLEW_VOLTAGE_RATE` attribute to this value.

Units: volts per second

Valid Range: 0.0 to 9.9e37

Default Value: 0.0 volts

NOTE:

If passed value is lower than zero then instrument will set maximum possible range (INFINITY).

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
--------	-------------

ERRORS:

BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggeredVoltageLevel***

```
ViStatus ciLxs_ConfigureTriggeredVoltageLevel
    (ViSession instrumentHandle, ViChar _VI_FAR phase[],
     ViReal64 triggeredLevel);
```

## Purpose

This function configures the AC rms output voltage level of the AC source.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type            ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## triggeredLevel

Variable Type            ViReal64

Pass the AC rms amplitude that the output waveform will be set to during a triggered step or pulse transient. The driver sets the `CILXS_ATTR_TRIGGERED_VOLTAGE_LEVEL` attribute to this value.

Units: volts

Valid Range:

0.0 to 300.0 (for sinewave)

Default Value: 0.0 volts

Note:

1) You cannot program a voltage that produces a higher volt-second on the output than a 300V rms sinewave.

2) The maximum peak voltage that the AC source can output is 425 V peak. This includes any combination of voltage and function shape values. Therefore, the maximum value that can be programmed depends on the peak-to-rms ratio of the selected waveform. For a sinewave, the maximum voltage that can be programmed is 300 V rms.

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings

---

3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

## ***ciLxs\_ConfigureTriggeredVoltageMode***

```
ViStatus ciLxs_ConfigureTriggeredVoltageMode (ViSession instrumentHandle,  
                                              ViChar _VI_FAR phaseName[],  
                                              ViInt32 transientMode);
```

### Purpose

This function configures the ac rms output voltage controlled during a triggered output transient.

### Parameter List

#### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

#### phaseName

Variable Type      ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

#### Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### transientMode

Variable Type      ViInt32

Specifies how the AC rms output voltage is controlled during a triggered output transient. The driver sets the CILXS\_ATTR\_TRIGGERED\_VOLTAGE\_MODE attribute to this value.

Valid Values:

- CILXS\_VAL\_TRIGGER\_MODE\_FIX - The voltage is unaffected by a triggered output transient.
- CILXS\_VAL\_TRIGGER\_MODE\_STEP - The voltage is programmed to the value set by `ciLxs_ConfigureTriggeredVoltageLevel` when a triggered transient occurs.
- CILXS\_VAL\_TRIGGER\_MODE\_PULSE - The voltage is changed to the value set by `ciLxs_ConfigureTriggeredVoltageLevel` for a duration determined by the `ciLxs_ConfigurePulse` function.
- CILXS\_VAL\_TRIGGER\_MODE\_LIST - The voltage is controlled by the voltage list when a triggered transient occurs.

Default Value:

CILXS\_VAL\_TRIGGER\_MODE\_FIX

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors

BFFC0000 to BFFCFFFF VXIPnP Driver Errors

---

***ciLxs\_ConfigureTriggerFrequencyList***

```
ViStatus ciLxs_ConfigureTriggerFrequencyList (ViSession instrumentHandle,  
                                              ViInt32 listSize,  
                                              ViReal64 _VI_FAR  
frequency[]);
```

## Purpose

This function configures the sequence of frequency list points.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## listSize

Variable Type      ViInt32

The number of list points.

## frequency

Variable Type      ViReal64[]

The sequence of frequency list points used to configure the AC Source.

The frequency points are given in the command parameters, which are separated by commas.

Unit: hertz

Valid Range:  
45.0 to 5000.0

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError`

function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

**ciLxs\_ConfigureTriggerFunctionList**

```

    ViStatus ciLxs_ConfigureTriggerFunctionList (ViSession instrumentHandle,
                                                ViChar _VI_FAR
functionsList[]);

```

Purpose

This function configures the sequence of the waveform shape entries.

Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

functionsList

Variable Type            ViChar[]

The sequence of function shapes list points.

The following values may be specified:

"SINUSOID", "SQUARE", "CSINUSOID" and user define function names.

The function names are separated by comma.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggerList***

```
ViStatus ciLxs_ConfigureTriggerList (ViSession instrumentHandle,
                                     ViInt32 repeatCount,
                                     ViInt32 stepMode, ViInt32 listSize,
                                     ViReal64 _VI_FAR dwellTimes[],
                                     ViInt32 _VI_FAR repeatTimes[]);
```

**Purpose**

This function configures how many times the AC source sequences through a list before that list is completed, specifies the time interval that each value (point) of a list is to remain in effect and determines if a trigger causes a list to advance only to its next point or to sequence through all of its points.

**Parameter List****instrumentHandle**

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

**repeatCount**

Variable Type      ViInt32

Pass the number of times that the list is executed before it is completed. The driver sets the `CILXS_ATTR_TRIGGER_LIST_COUNT` attribute to this value.

Valid Range: 1 to 2000000000 (2E9)

Default Value: 1

**NOTE:**

If passed value is lower than zero then instrument will set maximum possible range (INFINITY).

**stepMode**

Variable Type      ViInt32

Specifies how the list sequencing responds to triggers. The driver sets the `CILXS_ATTR_TRIGGER_LIST_MODE` attribute to this value.

**Valid Values:**

`CILXS_VAL_TRIGGER_LIST_STEP_ONCE` - causes the entire list to be output sequentially after the starting trigger, paced by its dwell

delays. As each dwell delay elapses, the next point is immediately output.

CILXS\_VAL\_TRIGGER\_LIST\_STEP\_AUTO - causes the list to advance only one point after each trigger. Triggers that arrive during a dwell delay are ignored.

Default Value: CILXS\_VAL\_TRIGGER\_LIST\_STEP\_AUTO

#### listSize

Variable Type            ViInt32

The size of the array of dwell times.

#### dwellTimes

Variable Type            ViReal64[]

This parameter sets the sequence of list dwell times. Each value represents the time in seconds that the output will remain at the particular list step point before completing the step. At the end of the dwell time, the output of the depends upon the following conditions:

- \* If step mode parameter has been set to CILXS\_VAL\_TRIGGER\_LIST\_STEP\_AUTO, the output automatically changes to the next point in the list.

- \* If step mode parameter has been set to CILXS\_VAL\_TRIGGER\_LIST\_STEP\_ONCE, the output remains at the present level until a trigger sequences the next point in the list.

The order in which the points are entered determines the sequence in which they are output when a list is triggered. Changing list data while a subsystem is in list mode generates an implied "abort".

#### repeatTimes

Variable Type            ViInt32[]

This parameter sets the sequence of list repeat times. Each value represents the time in seconds that the output will repeat at the particular list step point.

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError`

function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

## ***ciLxs\_ConfigureTriggerSource***

```
ViStatus ciLxs_ConfigureTriggerSource (ViSession instrumentHandle,  
                                       ViInt32 source);
```

### Purpose

This function configures the trigger source for the first sequence in generating a step, pulse, or list output.

### Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

source

Variable Type            ViInt32

Pass the trigger source to which you want the power supply to respond. The driver sets the `CILXS_ATTR_TRIGGER_SOURCE` attribute to this value.

#### Defined Values:

`CILXS_VAL_TRIG_IMMEDIATE` - The power supply does not wait for a trigger of any kind.

`CILXS_VAL_TRIG_EXTERNAL` - The power supply waits for a trigger on the external trigger input.

`CILXS_VAL_SOFTWARE_TRIG` - The power supply waits until you call the `ciLxs_SendSoftwareTrigger` function.

Default Value: `CILXS_VAL_TRIG_IMMEDIATE`

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggerTTLOutputList***

```

ViStatus ciLxs_ConfigureTriggerTTLOutputList (ViSession instrumentHandle,
                                              ViInt32 listSize,
                                              ViBoolean _VI_FAR
TTLOutput[]);

```

## Purpose

This function configures the sequence of Trigger Out list points.

## Parameter List

## instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## listSize

Variable Type          ViInt32

The number of list points.

## TTLOutput

Variable Type          ViBoolean[]

The sequence of Trigger Out list points. Each point which is set `VI_TRUE` will cause a pulse to be output at Trigger Out when that list step is reached. Those entries which are set `VI_FALSE` will not generate Trigger Out pulses. The order in which the list points are given determines the sequence in which Trigger Out pulses will be output when a list transient is triggered. Changing list data while a subsystem is in list mode generates an implied abort.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTriggerVoltageList***

```
ViStatus ciLxs_ConfigureTriggerVoltageList (ViSession instrumentHandle,
                                           ViChar _VI_FAR phase[],
                                           ViInt32 listSize,
                                           ViReal64 _VI_FAR voltage[]);
```

**Purpose**

This function configures the output voltage points in a list.

**Note:**

1) You cannot program a voltage that produces a higher volt-second on the output than a 300V rms sinewave.

**Parameter List****instrumentHandle**

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

**phase**

Variable Type      ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: "PHASE1"

**Notes:**

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

**listSize**

Variable Type      ViInt32

The number of list points.

voltage

Variable Type            ViReal64[]

The sequence of voltage list points used to configure the AC Source.

The voltage points are given in the command parameters, which are separated by commas. The order in which the points are entered determines the sequence in which the list will be output when a list transient is triggered.

Units: V (rms voltage)

Valid Range:  
 0.0 to 300.0 (for sinewaves)

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings

---

3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_ConfigureTrigSlewFrequencyList***

```
ViStatus ciLxs_ConfigureTrigSlewFrequencyList
        (ViSession instrumentHandle, ViInt32 listSize,
         ViReal64 _VI_FAR frequency[]);
```

## Purpose

This function configures the sequence of frequency slew list points.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## listSize

Variable Type      ViInt32

The number of list points.

## frequency

Variable Type      ViReal64[]

The sequence of frequency slew list points used to configure the AC Source.

The frequency points are given in the command parameters, which are separated by commas. The order in which the points are entered determines the sequence in which they are output when a list is triggered.

Units: HZ (Hertz) per second

Valid Range: 0.0 to 9.9e31

## NOTE:

If passed value is lower than zero then instrument will set maximum possible range (INFINITY).

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver

function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTrigSlewFrequencyMode***

```
ViStatus ciLxs_ConfigureTrigSlewFrequencyMode
(ViSession instrumentHandle, ViInt32 transientMode);
```

## Purpose

This function configures the frequency slew rate controlled during a triggered output transient.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## transientMode

Variable Type            ViInt32

Specifies how the frequency slew rate is controlled during a triggered output transient. The driver sets the `CILXS_ATTR_TRIGGERED_SLEW_FREQUENCY_RATE_MODE` attribute to this value.

## Valid Values:

- `CILXS_VAL_TRIGGER_MODE_FIX` - The frequency slew rate is unaffected by a triggered output transient.
- `CILXS_VAL_TRIGGER_MODE_STEP` - The frequency slew rate is programmed to the value set by `ciLxs_ConfigureTriggeredSlewFrequency` when a triggered transient occurs.
- `CILXS_VAL_TRIGGER_MODE_PULSE` - The frequency slew rate is changed to the value set by `ciLxs_ConfigureTriggeredSlewFrequency` for a duration determined by the `ciLxs_ConfigurePulse` function.
- `CILXS_VAL_TRIGGER_MODE_LIST` - The frequency slew rate is controlled by the frequency list when a triggered transient occurs.

## Default Value:

`CILXS_VAL_TRIGGER_MODE_FIX`

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the

error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ConfigureTrigSlewVoltageList***

```
ViStatus ciLxs_ConfigureTrigSlewVoltageList (ViSession instrumentHandle,
                                             ViChar _VI_FAR phase[],
                                             ViInt32 listSize,
                                             ViReal64 _VI_FAR voltage[]);
```

## Purpose

This function specifies the output offset slew points in a list.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type      ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: "PHASE1"

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## listSize

Variable Type      ViInt32

The number of list points.

voltage

Variable Type            ViReal64[]

The sequence of voltage slew list points used to configure the AC Source.

The slew points are given in the command parameters, which are separated by commas. The order in which the points are entered determines the sequence in which the list will be output when a list transient is triggered.

Units: V/S (volts per second)

Valid Range: 0.0 to 9.9e37

NOTE:

If passed value is lower than zero then instrument will set maximum possible range (INFINITY).

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
------------------------	-------------------

---

3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIIPnP	Driver Errors

---

## ***ciLxs\_ConfigureTrigSlewVoltageMode***

```
ViStatus ciLxs_ConfigureTrigSlewVoltageMode (ViSession instrumentHandle,  
                                             ViChar _VI_FAR phaseName[],  
                                             ViInt32 transientMode);
```

### Purpose

This function configures the output voltage slew rate controlled during a triggered output transient.

### Parameter List

#### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

#### phaseName

Variable Type      ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

#### Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### transientMode

Variable Type      ViInt32

Specifies how the output voltage slew rate is controlled during a triggered output transient. The driver sets the `CILXS_ATTR_TRIGGERED_SLEW_VOLTAGE_RATE_MODE` attribute to this value.

Valid Values:

- CILXS\_VAL\_TRIGGER\_MODE\_FIX - The slew rate is unaffected by a triggered output transient.
- CILXS\_VAL\_TRIGGER\_MODE\_STEP - The slew rate is programmed to the value set by `ciLxs_ConfigureTriggeredSlewVoltage` when a triggered transient occurs.
- CILXS\_VAL\_TRIGGER\_MODE\_PULSE - The slew rate is changed to the value set by `ciLxs_ConfigureTriggeredSlewVoltage` for a duration determined by the `ciLxs_ConfigurePulse` function.
- CILXS\_VAL\_TRIGGER\_MODE\_LIST -The slew rate is controlled by the voltage slew list when a triggered transient occurs.

Default Value:

CILXS\_VAL\_TRIGGER\_MODE\_FIX

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors

BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_ConfigureTTLTriggerOutput***

```
ViStatus ciLxs_ConfigureTTLTriggerOutput (ViSession instrumentHandle,
                                         ViBoolean state,
                                         ViInt32 source);
```

## Purpose

This function configures trigger out signal.

## Parameter List

## instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## state

Variable Type          ViBoolean

Pass the state of the AC source Trigger Out signal, which is available at a BNC connector on the rear of the iL Series units. The driver sets the CILXS\_ATTR\_OUTPUT\_TRIGGER\_ENABLED attribute to this value.

## Valid Values:

VI\_TRUE (1) - On  
VI\_FALSE (0) - Off (Default Value)

## source

Variable Type          ViInt32

Pass the signal source for the Trig Out signal. The driver sets the CILXS\_ATTR\_OUTPUT\_TRIGGER\_SOURCE attribute to this value.

## Valid Values:

CILXS\_VAL\_OUTPUT\_TRIGGER\_SOURCE\_BOT  
CILXS\_VAL\_OUTPUT\_TRIGGER\_SOURCE\_EOT  
CILXS\_VAL\_OUTPUT\_TRIGGER\_SOURCE\_LIST

Default Value: CILXS\_VAL\_OUTPUT\_TRIGGER\_SOURCE\_BOT

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

## ***ciLxs\_ConfigureVoltageLevel***

```
ViStatus ciLxs_ConfigureVoltageLevel (ViSession instrumentHandle,  
                                       ViChar _VI_FAR phase[],  
                                       ViReal64 level);
```

### Purpose

This function configures the AC RMS voltage level that the power supply attempts to generate.

### Parameter List

#### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

#### phase

Variable Type      ViChar[]

Pass the virtual phase name that you assign to the instrument in the Configuration Utility.

Virtual phase names are aliases for instrument-specific phase strings. The instrument-specific phase strings can differ from one instrument to another. Virtual phase names allow you to use and swap instruments without having to change the phase names in your source code. You assign a virtual phase name to an instrument-specific phase through the Configuration Utility. This control accepts virtual phase names you have assigned to the specific instrument you are using. It also accepts the instrument-specific phase names.

Default Value: ""

#### Notes:

(1) You can specify the phase name as a string variable or as a literal enclosed in double quotes.

#### level

Variable Type      ViReal64

Pass the AC RMS voltage you want the power supply to attempt to generate for the specified phase. The driver sets the `CILXS_ATTR_VOLTAGE_LEVEL` attribute to this value.

Units: volts

Valid Range:

0.0 to 300.0 (for sinewaves)

Default Value: 1.0 volts

Note:

1) You cannot program a voltage that produces a higher volt-second on the output than a 300V rms sinewave.

2) The maximum peak voltage that the AC source can output is 425 V peak. This includes any combination of voltage and function shape values. Therefore, the maximum value that can be programmed depends on the peak-to-rms ratio of the selected waveform. For a sinewave, the maximum voltage that can be programmed is 300 V rms.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings

---

3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_ConfigureWaveform***

```
ViStatus ciLxs_ConfigureWaveform (ViSession instrumentHandle,
                                  ViChar _VI_FAR function[],
                                  ViReal64 frequency);
```

## Purpose

This function configures the shape of waveform, which AC power produce on the output.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## function

Variable Type            ViChar[]

Pass the shape of the output voltage waveform. The driver sets the `CILXS_ATTR_FUNCTION` attribute to this value.

## Valid Values:

"SINUSOID", "SQUARE", "CSINUSOID" or user defined function name

If you specify "CSINUSOID" (clipped sinusoid) function, you can specify clipping level with `ciLxs_ConfigureClippingLevel` function.

Default Value: "SINUSOID"

## Note:

1) Before programming a different waveform shape, the output voltage should be programmed to zero volts. After the shape is changed, the voltage maybe programmed to the desired value.

## frequency

Variable Type            ViReal64

Pass the frequency of the output waveform. The driver sets the `CILXS_ATTR_FREQUENCY` attribute to this value.

Units: hertz

## Valid Range:

45.0 to 5000.0

Default Value: 60.0 hertz

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_Disable***

```
ViStatus ciLxs_Disable (ViSession instrumentHandle);
```

Purpose

This function places the instrument in a quiescent state where it has minimal or no impact on the system to which it is connected.

Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
WARNINGS:	
None	
ERRORS:	
BFFA4001	Histogram is not enabled.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the

particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviScope Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviScope Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_error\_message***

```
ViStatus ciLxs_error_message (ViSession instrumentHandle,
                             ViStatus errorCode,
                             ViChar _VI_FAR errorMessage[]);
```

## Purpose

This function converts a status code returned by an instrument driver function into a user-readable string.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

You can pass `VI_NULL` for this parameter. This is useful when one of the initialize functions fail.

Default Value: `VI_NULL`

## errorCode

Variable Type            ViStatus

Pass the Status parameter that is returned from any of the instrument driver functions.

Default Value: `0 (VI_SUCCESS)`

## IviDCPwr Status Codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

## IviDCPwr Status Codes:

Status	Description
-----	
ERRORS:	
BFFA0001	Instrument error. Call <code>ciLxs_error_query</code> .
BFFA0002	Cannot open file.
BFFA0003	Error reading from file.
BFFA0004	Error writing to file.
BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

## IVI Engine Status Codes:

Status	Description
-----	
ERRORS:	
BFFA0001	Instrument error. Call <code>ciLxs_error_query</code> .
BFFA0002	Cannot open file.
BFFA0003	Error reading from file.
BFFA0004	Error writing to file.
BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

## ERRORS:

BFFA0001	Instrument error. Call <code>ciLxs_error_query</code> .
BFFA0002	Cannot open file.
BFFA0003	Error reading from file.
BFFA0004	Error writing to file.
BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0002	Cannot open file.
BFFA0003	Error reading from file.
BFFA0004	Error writing to file.
BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0003	Error reading from file.
BFFA0004	Error writing to file.
BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0004	Error writing to file.
BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0005	Driver module file not found.
BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0006	Cannot open driver module file for reading.
BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0007	Driver module has invalid file format or invalid data.
BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0008	Driver module contains undefined references.
BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA0009	Cannot find function in driver module.
BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA000A	Failure loading driver module.
BFFA000B	Invalid path name.

BFFA000B	Invalid path name.
----------	--------------------

BFFA000C	Invalid attribute.
BFFA000D	IVI attribute is not writable.
BFFA000E	IVI attribute is not readable.
BFFA000F	Invalid parameter.
BFFA0010	Invalid value.
BFFA0011	Function not supported.
BFFA0012	Attribute not supported.
BFFA0013	Value not supported.
BFFA0014	Invalid type.
BFFA0015	Types do not match.
BFFA0016	Attribute already has a value waiting to be updated.
BFFA0017	Specified item already exists.
BFFA0018	Not a valid configuration.
BFFA0019	Requested item does not exist or value not available.
BFFA001A	Requested attribute value not known.
BFFA001B	No range table.
BFFA001C	Range table is invalid.
BFFA001D	Object or item is not initialized.
BFFA001E	Non-interchangeable behavior.
BFFA001F	No channel table has been built for the session.
BFFA0020	Channel name specified is not valid.
BFFA0021	Unable to allocate system resource.
BFFA0022	Permission to access file was denied.
BFFA0023	Too many files are already open.
BFFA0024	Unable to create temporary file in target directory.
BFFA0025	All temporary filenames already used.
BFFA0026	Disk is full.
BFFA0027	Cannot find configuration file on disk.
BFFA0028	Cannot open configuration file.
BFFA0029	Error reading configuration file.
BFFA002A	Invalid ViInt32 value in configuration file.
BFFA002B	Invalid ViReal64 value in configuration file.
BFFA002C	Invalid ViBoolean value in configuration file.
BFFA002D	Entry missing from configuration file.
BFFA002E	Initialization failed in driver DLL.
BFFA002F	Driver module has unresolved external reference.
BFFA0030	Cannot find CVI Run-Time Engine.
BFFA0031	Cannot open CVI Run-Time Engine.
BFFA0032	CVI Run-Time Engine has invalid format.
BFFA0033	CVI Run-Time Engine is missing required function(s).
BFFA0034	CVI Run-Time Engine initialization failed.
BFFA0035	CVI Run-Time Engine has unresolved external reference.
BFFA0036	Failure loading CVI Run-Time Engine.
BFFA0037	Cannot open DLL for read exports.
BFFA0038	DLL file is corrupt.
BFFA0039	No DLL export table in DLL.
BFFA003A	Unknown attribute name in default configuration file.
BFFA003B	Unknown attribute value in default configuration file.
BFFA003C	Memory pointer specified is not known.
BFFA003D	Unable to find any channel strings.
BFFA003E	Duplicate channel string.
BFFA003F	Duplicate virtual channel name.
BFFA0040	Missing virtual channel name.
BFFA0041	Bad virtual channel name.
BFFA0042	Unassigned virtual channel name.
BFFA0043	Bad virtual channel assignment.
BFFA0044	Channel name required.

BFFA0045 Channel name not allowed.  
 BFFA0046 Attribute not valid for channel.  
 BFFA0047 Attribute must be channel based.  
 BFFA0048 Channel already excluded.  
 BFFA0049 Missing option name (nothing before the '=').  
 BFFA004A Missing option value (nothing after the '=').  
 BFFA004B Bad option name.  
 BFFA004C Bad option value.  
 BFFA004D Operation only valid on a class driver session.  
 BFFA004E "ivi.ini" filename is reserved.  
 BFFA004F Duplicate run-time configuration entry.  
 BFFA0050 Index parameter is one-based.  
 BFFA0051 Index parameter is too high.  
 BFFA0052 Attribute is not cacheable.  
 BFFA0053 You cannot export a ViAddr attribute to the end-user.  
 BFFA0054 Bad channel string in channel string list.  
 BFFA0055 Bad prefix name in default configuration file.

#### VISA Status Codes:

Status	Description
-----	
WARNINGS:	
3FFF0002	Event enabled for one or more specified mechanisms.
3FFF0003	Event disabled for one or more specified mechanisms.
3FFF0004	Successful, but queue already empty.
3FFF0005	Specified termination character was read.
3FFF0006	Number of bytes transferred equals input count.
3FFF0077	Configuration non-existent or could not be loaded.
3FFF007D	Open successful, but the device not responding.
3FFF0080	Wait successful, but more event objects available.
3FFF0082	Specified object reference is uninitialized.
3FFF0084	Attribute value not supported.
3FFF0085	Status code could not be interpreted.
3FFF0088	Specified I/O buffer type not supported.
3FFF0098	Successful, but invoke no handlers for this event.
3FFF0099	Successful but session has nested shared locks.
3FFF009A	Successful but session has nested exclusive locks.
3FFF009B	Successful but operation not asynchronous.
ERRORS:	
BFFF0000	Unknown system error (miscellaneous error).
BFFF000E	Session or object reference is invalid.
BFFF000F	Resource is locked.
BFFF0010	Invalid expression specified for search.
BFFF0011	Resource is not present in the system.
BFFF0012	Invalid resource reference specified. Parsing error.
BFFF0013	Invalid access mode.
BFFF0015	Timeout expired before operation completed.
BFFF0016	Unable to deallocate session data structures.
BFFF001B	Specified degree is invalid.
BFFF001C	Specified job identifier is invalid.
BFFF001D	Attribute is not supported by the referenced object.
BFFF001E	Attribute state not supported by the referenced object.
BFFF001F	Specified attribute is read-only.
BFFF0020	Lock type lock not supported by this resource.
BFFF0021	Invalid access key.
BFFF0026	Specified event type not supported by the resource.

#### WARNINGS:

3FFF0002 Event enabled for one or more specified mechanisms.  
 3FFF0003 Event disabled for one or more specified mechanisms.  
 3FFF0004 Successful, but queue already empty.  
 3FFF0005 Specified termination character was read.  
 3FFF0006 Number of bytes transferred equals input count.  
 3FFF0077 Configuration non-existent or could not be loaded.  
 3FFF007D Open successful, but the device not responding.  
 3FFF0080 Wait successful, but more event objects available.  
 3FFF0082 Specified object reference is uninitialized.  
 3FFF0084 Attribute value not supported.  
 3FFF0085 Status code could not be interpreted.  
 3FFF0088 Specified I/O buffer type not supported.  
 3FFF0098 Successful, but invoke no handlers for this event.  
 3FFF0099 Successful but session has nested shared locks.  
 3FFF009A Successful but session has nested exclusive locks.  
 3FFF009B Successful but operation not asynchronous.

#### ERRORS:

BFFF0000 Unknown system error (miscellaneous error).  
 BFFF000E Session or object reference is invalid.  
 BFFF000F Resource is locked.  
 BFFF0010 Invalid expression specified for search.  
 BFFF0011 Resource is not present in the system.  
 BFFF0012 Invalid resource reference specified. Parsing error.  
 BFFF0013 Invalid access mode.  
 BFFF0015 Timeout expired before operation completed.  
 BFFF0016 Unable to deallocate session data structures.  
 BFFF001B Specified degree is invalid.  
 BFFF001C Specified job identifier is invalid.  
 BFFF001D Attribute is not supported by the referenced object.  
 BFFF001E Attribute state not supported by the referenced object.  
 BFFF001F Specified attribute is read-only.  
 BFFF0020 Lock type lock not supported by this resource.  
 BFFF0021 Invalid access key.  
 BFFF0026 Specified event type not supported by the resource.

BFFF0027 Invalid mechanism specified.  
 BFFF0028 A handler was not installed.  
 BFFF0029 Handler reference either invalid or was not installed.  
 BFFF002A Specified event context invalid.  
 BFFF002D Event queue for specified type has overflowed.  
 BFFF002F Event type must be enabled in order to receive.  
 BFFF0030 User abort during transfer.  
 BFFF0034 Violation of raw write protocol during transfer.  
 BFFF0035 Violation of raw read protocol during transfer.  
 BFFF0036 Device reported output protocol error during transfer.  
 BFFF0037 Device reported input protocol error during transfer.  
 BFFF0038 Bus error during transfer.  
 BFFF0039 Unable to queue asynchronous operation.  
 BFFF003A Unable to start operation because setup is invalid.  
 BFFF003B Unable to queue the asynchronous operation.  
 BFFF003C Insufficient resources to perform memory allocation.  
 BFFF003D Invalid buffer mask specified.  
 BFFF003E I/O error.  
 BFFF003F Format specifier invalid.  
 BFFF0041 Format specifier not supported.  
 BFFF0042 Trigger line is currently in use.  
 BFFF004A Service request not received for the session.  
 BFFF004E Invalid address space specified.  
 BFFF0051 Invalid offset specified.  
 BFFF0052 Invalid access width specified.  
 BFFF0054 Offset not accessible from this hardware.  
 BFFF0055 Source and destination widths are different.  
 BFFF0057 Session not currently mapped.  
 BFFF0059 Previous response still pending.  
 BFFF005F No listeners condition detected.  
 BFFF0060 Interface not currently the controller in charge.  
 BFFF0061 Interface not the system controller.  
 BFFF0067 Session does not support this operation.  
 BFFF006A A parity error occurred during transfer.  
 BFFF006B A framing error occurred during transfer.  
 BFFF006C An overrun error occurred during transfer.  
 BFFF0070 Offset not properly aligned for operation access width.  
 BFFF0071 Specified user buffer not valid.  
 BFFF0072 Resource valid, but VISA cannot access it.  
 BFFF0076 Width not supported by this hardware.  
 BFFF0078 Invalid parameter value, parameter unknown.  
 BFFF0079 Invalid protocol.  
 BFFF007B Invalid window size.  
 BFFF0080 Session currently contains a mapped window.  
 BFFF0081 Operation not implemented.  
 BFFF0083 Invalid length.  
 BFFF0091 Invalid mode.  
 BFFF009C Session did not have a lock on the resource.  
 BFFF009D The device does not export any memory.  
 BFFF009E VISA-required code library not located or not loaded.

VXIPnP Driver Status Codes:

Status	Description
-----	
WARNINGS:	
3FFC0101	Instrument does not have ID Query capability.
3FFC0102	Instrument does not have Reset capability.

-----

3FFC0103 Instrument does not have Self-Test capability.  
 3FFC0104 Instrument does not have Error Query capability.  
 3FFC0105 Instrument does not have Revision Query capability.

ERRORS:

BFFC0001 Parameter 1 out of range, or error trying to set it.  
 BFFC0002 Parameter 2 out of range, or error trying to set it.  
 BFFC0003 Parameter 3 out of range, or error trying to set it.  
 BFFC0004 Parameter 4 out of range, or error trying to set it.  
 BFFC0005 Parameter 5 out of range, or error trying to set it.  
 BFFC0006 Parameter 6 out of range, or error trying to set it.  
 BFFC0007 Parameter 7 out of range, or error trying to set it.  
 BFFC0008 Parameter 8 out of range, or error trying to set it.  
 BFFC0011 Instrument failed the ID Query.  
 BFFC0012 Invalid response from instrument.

errorMessage

Variable Type ViChar[]

Returns the user-readable message string that corresponds to the status code you specify.

You must pass a ViChar array with at least 256 bytes.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional

status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_error\_query***

```
ViStatus ciLxs_error_query (ViSession instrumentHandle,  
                           ViPInt32 errorCode,  
                           ViChar _VI_FAR errorMessage[]);
```

## Purpose

This function reads an error code and a message from the instrument's error queue.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

## errorCode

Variable Type      ViInt32 (passed by reference)

Returns the error code read from the instrument's error queue.

## errorMessage

Variable Type      ViChar[]

Returns the error message string read from the instrument's error message queue.

You must pass a ViChar array with at least 256 bytes.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

## ***ciLxs\_Fetch***

```
ViStatus ciLxs_Fetch (ViSession instrumentHandle,  
                    ViChar _VI_FAR phase[], ViInt32 measurementType,  
                    ViPReal64 measurement);
```

### Purpose

This function takes a single measurement on the phase you specify.

### Parameter List

#### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

#### phase

Variable Type      ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

#### Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### measurementType

Variable Type      ViInt32

Pass the measurement you want the power supply to take.

#### Defined Values:

CILXS\_VAL\_MEASURE\_AC\_VOLTAGE\_RMS - ac rms voltage

CILXS\_VAL\_MEASURE\_AC\_CURRENT\_RMS - ac rms current  
 CILXS\_VAL\_MEASURE\_CURRENT\_MAXIMUM - peak current  
 CILXS\_VAL\_MEASURE\_CURRENT\_CRESTFACTOR - current crestfactor  
 CILXS\_VAL\_MEASURE\_AC\_POWER - real power  
 CILXS\_VAL\_MEASURE\_AC\_APPARENT\_POWER - apparent power  
 CILXS\_VAL\_MEASURE\_AC\_REACTIVE\_POWER - reactive power  
 CILXS\_VAL\_MEASURE\_AC\_TOTAL\_POWER - total power  
 CILXS\_VAL\_MEASURE\_AC\_POWER\_FACTOR - output power factor  
 CILXS\_VAL\_MEASURE\_NEUTRAL\_AC\_CURRENT\_RMS - neutral ac rms current  
 (3-phase only)  
  
 CILXS\_VAL\_MEASURE\_FREQUENCY - output frequency  
 CILXS\_VAL\_MEASURE\_PHASE - output phase

Default Value: CILXS\_VAL\_MEASURE\_AC\_VOLTAGE\_RMS

measurement

Variable Type            ViReal64 (passed by reference)

Returns the measured value.

Units: volts (for voltage measurement)  
       amps (for current measurement)  
       watts (for power measurement)  
       voltamperes (for apparent power measurement)  
       voltamperes reactive (for reactive power measurement)  
       hertz (for current frequency)

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_FetchArray***

```
ViStatus ciLxs_FetchArray (ViSession instrumentHandle,
                          ViChar _VI_FAR phase[],
                          ViInt32 measurementType, ViInt32 arraySize,
                          ViReal64 _VI_FAR measurement[],
                          ViPInt32 number_of_measurements);
```

**Purpose**

This function takes a harmonic measurement on the phase you specify.

**Parameter List****instrumentHandle**

Variable Type          ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

**phase**

Variable Type          ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: "PHASE1"

**Notes:**

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

**measurementType**

Variable Type          ViInt32

Pass the type of measurement to retrieve.

Defined Values:

CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE CILXS\_VAL\_HARMONIC\_VOLTAGE\_PHASE  
 CILXS\_VAL\_HARMONIC\_CURRENT\_AMPLITUDE CILXS\_VAL\_HARMONIC\_CURRENT\_PHASE  
 CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_AMPLITUDE  
 CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_PHASE  
 CILXS\_VAL\_MEASURE\_DC\_VOLTAGE

CILXS\_VAL\_MEASURE\_DC\_CURRENT  
 CILXS\_VAL\_MEASURE\_NEUTRAL\_DC\_CURRENT

Default Value: CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE

arraySize

Variable Type ViInt32

Specifies size of measurement array. Harmonic measurements require an array of size 50. All other measurements require an array of size 4096.

measurement

Variable Type ViReal64[]

Returns the measurements retrieved from the AC Source.

number\_of\_measurements

Variable Type ViInt32 (passed by reference)

Returns the number of valid values in measurement array.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	

BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_FetchHarmonic***

```
ViStatus ciLxs_FetchHarmonic (ViSession instrumentHandle,
                             ViChar _VI_FAR phase[], ViInt32 harmonic,
                             ViInt32 measurementType,
                             ViPReal64 measurement);
```

## Purpose

This function takes a harmonic measurement on the phase you specify.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type            ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## harmonic

Variable Type            ViInt32

Pass the desired harmonic number. Queries sent with a value of 0 return the dc component. A value of 1 returns the fundamental output frequency. Harmonic orders can be queried up to the fundamental measurement bandwidth of the measurement system,

which is 12.6kHz. Thus the maximum harmonic that can be measured is dependent on the output frequency. Any harmonics that represent frequencies greater than 12.6kHz are returned as 0.

Default Value: 1

measurementType

Variable Type            ViInt32

Pass the measurement you want the power supply to take.

Defined Values:

- CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE - Voltage Amplitude
- CILXS\_VAL\_HARMONIC\_VOLTAGE\_PHASE     - Voltage Phase
- CILXS\_VAL\_HARMONIC\_CURRENT\_AMPLITUDE - Current Amplitude
- CILXS\_VAL\_HARMONIC\_CURRENT\_PHASE     - Current Phase
- CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_AMPLITUDE - Neutral Current Amplitude
- CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_PHASE     - Neutral Current Phase

Default Value: CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE

measurement

Variable Type            ViReal64 (passed by reference)

Returns the measured value.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code	Types
-----		
3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

***ciLxs\_GetAttributeViBoolean***

```
ViStatus ciLxs_GetAttributeViBoolean (ViSession instrumentHandle,
                                     ViChar _VI_FAR channelName[],
                                     ViAttr attributeID,
                                     ViPBoolean attributeValue);
```

## Purpose

This function queries the value of a ViBoolean attribute.

You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

## Parameter List

**instrumentHandle**

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

**channelName**

Variable Type            ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be retrieved. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViBoolean type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

#### attributeValue

Variable Type            ViBoolean (passed by reference)

Returns the current value of the attribute. Pass the address of a ViBoolean variable.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_GetAttributeViInt32***

```
ViStatus ciLxs_GetAttributeViInt32 (ViSession instrumentHandle,
                                   ViChar _VI_FAR channelName[],
                                   ViAttr attributeID,
                                   ViPInt32 attributeValue);
```

## Purpose

This function queries the value of a ViInt32 attribute.

You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

## channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be retrieved. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the

instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box.

Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

#### attributeValue

Variable Type            ViInt32 (passed by reference)

Returns the current value of the attribute. Pass the address of a ViInt32 variable.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

## ***ciLxs\_GetAttributeViReal64***

```
ViStatus ciLxs_GetAttributeViReal64 (ViSession instrumentHandle,  
                                     ViChar _VI_FAR channelName[],  
                                     ViAttr attributeID,  
                                     ViPReal64 attributeValue);
```

### Purpose

This function queries the value of a ViReal64 attribute.

You can use this function to get the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

### Parameter List

#### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

#### channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be retrieved. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViReal64 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

#### attributeValue

Variable Type            ViReal64 (passed by reference)

Returns the current value of the attribute. Pass the address of a ViReal64 variable.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_GetAttributeViSession***

```
ViStatus ciLxs_GetAttributeViSession (ViSession instrumentHandle,
                                     ViChar _VI_FAR channelName[],
                                     ViAttr attributeID,
                                     ViPSession attributeValue);
```

## Purpose

This function queries the value of a ViSession attribute.

You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## channelName

Variable Type            ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be retrieved. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an

instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the

instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViSession type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

#### attributeValue

Variable Type            ViSession (passed by reference)

Returns the current value of the attribute. Pass the address of a ViSession variable.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_GetAttributeViString***

```
ViStatus ciLxs_GetAttributeViString (ViSession instrumentHandle,
                                     ViChar _VI_FAR channelName[],
                                     ViAttr attributeID,
                                     ViInt32 bufferSize,
                                     ViChar _VI_FAR attributeValue[]);
```

**Purpose**

This function queries the value of a ViString attribute.

You can use this function to get the values of instrument- specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid.

You must provide a ViChar array to serve as a buffer for the value. You pass the number of bytes in the buffer as the Buffer Size parameter. If the current value of the attribute, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you want to call this function just to get the required buffer size, you can pass 0 for the Buffer Size and VI\_NULL for the Attribute Value buffer.

If you want the function to fill in the buffer regardless of the number of bytes in the value, pass a negative number for the Buffer Size parameter.

**Parameter List**

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or

ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be retrieved. If the attribute is not channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

attributeID

Variable Type      ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViString type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.

- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

#### bufferSize

Variable Type            ViInt32

Pass the number of bytes in the ViChar array you specify for the Attribute Value parameter.

If the current value of the attribute, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI\_NULL for the Attribute Value buffer parameter.

Default Value: 512

#### attributeValue

Variable Type            ViChar[]

The buffer in which the function returns the current value of the attribute. The buffer must be of type ViChar and have at least as many bytes as indicated in the Buffer Size parameter.

If the current value of the attribute, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you specify 0 for the Buffer Size parameter, you can pass VI\_NULL for this parameter.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_GetError***

```
ViStatus ciLxs_GetError (ViSession instrumentHandle, ViPStatus code,
                        ViInt32 bufferSize,
                        ViChar _VI_FAR description[]);
```

**Purpose**

This function retrieves and then clears the IVI error information for the session or the current execution thread. One exception exists: If the BufferSize parameter is 0, 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.

If the user specifies a valid IVI session for the InstrumentHandle parameter, Get Error retrieves and then clears the error information for the session. If the user passes VI\_NULL for the InstrumentHandle parameter, this function retrieves and then clears the error information for the current execution thread. If the InstrumentHandle 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 ciLxs\_GetError or ciLxs\_ClearError.

**Parameter List****instrumentHandle**

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

**code**

Variable Type            ViStatus (passed by reference)

Returns the error code for the session or execution thread.

If you pass 0 for the Buffer Size, you can pass VI\_NULL for this parameter.

**bufferSize**

Variable Type            ViInt32

Pass the number of bytes in the ViChar array you specify for the Description parameter.

If the error description, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies BufferSize - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI\_NULL for the Description buffer parameter.

Default Value: None

description

Variable Type      ViChar[]

Returns the error description for the IVI session or execution thread. If there is no description, the function returns an empty string.

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the error description, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass 0 for the Buffer Size, you can pass VI\_NULL for this parameter.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the

error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
WARNINGS:	
None	
ERRORS:	
BFFA4001	Histogram is not enabled.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	
3FFA2000 to 3FFA3FFF	IviScope Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviScope Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_GetNextCoercionRecord***

```
ViStatus ciLxs_GetNextCoercionRecord (ViSession instrumentHandle,
                                       ViInt32 bufferSize,
                                       ViChar _VI_FAR coercionRecord[]);
```

**Purpose**

This function returns the coercion information associated with the IVI session. This function retrieves and clears the oldest instance in which the instrument driver coerced a value you specified to another value.

If you set the CILXS\_ATTR\_RECORD\_COERCIONS attribute to VI\_TRUE, the instrument driver keeps a list of all coercions it makes on ViInt32 or ViReal64 values you pass to instrument driver functions. You use this function to retrieve information from that list.

If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI\_NULL for the Coercion Record buffer parameter.

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

**Parameter List****instrumentHandle**

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init function. The handle identifies a particular instrument session.

Default Value:    None

**bufferSize**

Variable Type      ViInt32

Pass the number of bytes in the ViChar array you specify for the Coercion Record parameter.

If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an

ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI\_NULL for the Coercion Record buffer parameter.

Default Value: None

#### coercionRecord

Variable Type            ViChar[]

Returns the next coercion record for the IVI session. If there are no coercion records, the function returns an empty string.

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the next coercion record string, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

This parameter returns an empty string if no coercion records remain for the session.

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
WARNINGS:	
none	
ERRORS:	
none	

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviDCPwr Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviDCPwr Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_GetNextInterchangeWarning***

```

ViStatus ciLxs_GetNextInterchangeWarning (ViSession instrumentHandle,
                                          ViInt32 bufferSize,
                                          ViChar _VI_FAR
interchangeWarning[]);

```

**Purpose**

This function returns the interchangeability warnings associated with the IVI session. It retrieves and clears the oldest instance in which the class driver recorded an interchangeability warning. Interchangeability warnings indicate that using your application with a different instrument might cause different behavior. You use this function to retrieve interchangeability warnings.

The driver performs interchangeability checking when the CILXS\_ATTR\_INTERCHANGE\_CHECK attribute is set to VI\_TRUE.

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

In general, the instrument driver generates interchangeability warnings when an attribute that affects the behavior of the instrument is in a state that you did not specify.

**Parameter List****instrumentHandle**

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

**bufferSize**

Variable Type	ViInt32
---------------	---------

Pass the number of bytes in the ViChar array you specify for the Interchange Warning parameter.

If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the

buffer regardless of the number of bytes in the value.

If you pass 0, you can pass VI\_NULL for the Interchange Warning buffer parameter.

Default Value: None

#### interchangeWarning

Variable Type            ViChar[]

Returns the next interchange warning for the IVI session. If there are no interchange warnings, the function returns an empty string.

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

This parameter returns an empty string if no interchangeability warnings remain for the session.

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetErrorInfo` function. To clear the error information from the driver, call the `ciLxs_ClearErrorInfo` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings

Negative Values                      Errors

This driver defines the following status codes:

Status	Description
-----	
WARNINGS:	
none	
ERRORS:	
none	

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	
3FFA2000 to 3FFA3FFF	IviDCPwr Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviDCPwr Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_GetPhaseName***

```
ViStatus ciLxs_GetPhaseName (ViSession instrumentHandle, ViInt32 index,  
                             ViInt32 bufferSize,  
                             ViChar _VI_FAR phaseName[]);
```

## Purpose

This function returns the highest-level phase name that corresponds to the specific driver phase string that is in the phase table at an index you specify. By passing 0 for the buffer size, the caller can ascertain the buffer size required to get the entire phase name string and then call the function again with a sufficiently large buffer.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## index

Variable Type      ViInt32

A 1-based index into the phase table.

Default Value: 1

## bufferSize

Variable Type      ViInt32

Pass the number of bytes in the ViChar array you specify for the Channel Name parameter.

If the channel name, including the terminating NUL byte, contains more bytes than you indicate in this parameter, the function copies `BufferSize - 1` bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.

If you pass 0, you can pass `VI_NULL` for the Channel Name buffer

parameter.

Default Value: 8

phaseName

Variable Type ViChar[]

Returns the highest-level phase name that corresponds to the specific driver phase string that is in the phase table at an index you specify.

The buffer must contain at least as many elements as the value you specify with the Buffer Size parameter. If the phase name description, including the terminating NUL byte, contains more bytes than you indicate with the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If you pass 0 for the Buffer Size, you can pass VI\_NULL for this parameter.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

If the current value of the return buffer, including the terminating NUL byte, is larger than the size you indicate in the Buffer Size parameter, the function copies Buffer Size - 1 bytes into the buffer, places an ASCII NUL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status Description

---

**WARNINGS:**

None

**ERRORS:**

None

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviScope Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviScope Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_init***

```
ViStatus ciLxs_init (ViRsrc resourceName, ViBoolean IDQuery,
                   ViBoolean resetDevice, ViPSession instrumentHandle,
                   ViInt32 baudRate);
```

## Purpose

This function performs the following initialization actions:

- Creates a new IVI instrument driver session.
- Opens a session to the specified device using the interface and address you specify for the Resource Name parameter.
- If the ID Query parameter is set to VI\_TRUE, this function queries the instrument ID and checks that it is valid for this instrument driver.
- If the Reset parameter is set to VI\_TRUE, this function resets the instrument to a known state.
- Sends initialization commands to set the instrument to the state necessary for the operation of the instrument driver.
- Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Note: This function creates a new session each time you invoke it. Although you can open more than one IVI session for the same resource, it is best not to do so. You can use the same session in multiple program threads. You can use the `ciLxs_LockSession` and `ciLxs_UnlockSession` functions to protect sections of code that require exclusive access to the resource.

## Parameter List

resourceName

Variable Type      ViRsrc

Pass the resource name of the device to initialize.

Refer to the following table below for the exact grammar to use for this parameter. Optional fields are shown in square brackets ([]).

Interface	Syntax
GPIB	GPIB[board]::<primary address> [::secondary address]::INSTR
Serial	ASRL<port>::INSTR

Use the GPIB keyword for GPIB instruments. Use the ASRL keyword for serial instruments.

If you do not specify a value for an optional field, the following values are used:

Optional Field	Value
board	0
secondary address	none (31)

The following table contains example valid resource names.

Resource Name	Description
"GPIB::22::INSTR"	GPIB board 0, primary address 22 no secondary address
"GPIB::22::5::INSTR"	GPIB board 0, primary address 22 secondary address 5
"GPIB1::22::5::INSTR"	GPIB board 1, primary address 22 secondary address 5
"ASRL2::INSTR"	COM port 2

Default Value: "GPIB::1::INSTR"

IDQuery

Variable Type ViBoolean

Specify whether you want the instrument driver to perform an ID Query.

Valid Range:

- VI\_TRUE (1) - Perform ID Query (Default Value)
- VI\_FALSE (0) - Skip ID Query

When you set this parameter to VI\_TRUE, the driver verifies that the instrument you initialize is a type that this driver supports.

Circumstances can arise where it is undesirable to send an ID Query command string to the instrument. When you set this parameter to VI\_FALSE, the function initializes the instrument without performing an ID Query.

resetDevice

Variable Type ViBoolean

Specify whether you want the to reset the instrument during the initialization procedure.

Valid Range:

- VI\_TRUE (1) - Reset Device (Default Value)
- VI\_FALSE (0) - Don't Reset

`instrumentHandle`

Variable Type            `ViSession` (passed by reference)

Returns a `ViSession` handle that you use to identify the instrument in all subsequent instrument driver function calls.

## Notes:

(1) This function creates a new session each time you invoke it. This is useful if you have multiple physical instances of the same type of instrument.

(2) Avoid creating multiple concurrent sessions to the same physical instrument. Although you can create more than one IVI session for the same resource, it is best not to do so. A better approach is to use the same IVI session in multiple execution threads. You can use

functions `ciLxs_LockSession` and `ciLxs_UnlockSession` to protect sections of code that require exclusive access to the resource.

`baudRate`

Variable Type            `ViInt32`

Specify the baud rate of the serial port.

## Baud Rate Ranges:

9600  
19200  
38400  
57600  
115200

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings

Negative Values                      Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:

BFFA1001    The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings
3FFF0000 to 3FFFFFFF	VISA          Warnings
3FFC0000 to 3FFCFFFF	VXIPnP      Driver Warnings
-----	
BFFA0000 to BFFA1FFF	IVI            Errors
BFFF0000 to BFFFFFFF	VISA          Errors
BFFC0000 to BFFCFFFF	VXIPnP      Driver Errors

**ciLxs\_InitiateAcquisition**

```
ViStatus ciLxs_InitiateAcquisition (ViSession instrumentHandle);
```

Purpose

This function initiates acquisition.

Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
------------------------	-------------------

---

3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_InitiateTransient***

```
ViStatus ciLxs_InitiateTransient (ViSession instrumentHandle);
```

Purpose

This function initiates transient.

Note:

1) During pulse or list transient, CILXS\_ATTR\_VOLTAGE\_LEVEL, CILXS\_ATTR\_PHASE\_ANGLE, CILXS\_ATTR\_FREQUENCY, CILXS\_ATTR\_FUNCTION, CILXS\_ATTR\_SLEW\_VOLTAGE\_RATE, CILXS\_ATTR\_SLEW\_FREQUENCY\_RATE attributes can returns inconsistent values with actual output. Instrument does not monitor output during transient.

Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_InitWithOptions***

```
ViStatus ciLxs_InitWithOptions (ViRsrc resourceName, ViBoolean IDQuery,
                               ViBoolean resetDevice,
                               ViConstString optionString,
                               ViPSession instrumentHandle,
                               ViInt32 baudRate);
```

## Purpose

This function performs the following initialization actions:

- Creates a new IVI instrument driver and optionally sets the initial state of the following session attributes:

```
CILXS_ATTR_RANGE_CHECK
CILXS_ATTR_QUERY_INSTR_STATUS
CILXS_ATTR_CACHE
CILXS_ATTR_SIMULATE
CILXS_ATTR_RECORD_COERCIONS
```

- Opens a session to the specified device using the interface and address you specify for the Resource Name parameter.
- If the ID Query parameter is set to VI\_TRUE, this function queries the instrument ID and checks that it is valid for this instrument driver.
- If the Reset parameter is set to VI\_TRUE, this function resets the instrument to a known state.
- Sends initialization commands to set the instrument to the state necessary for the operation of the instrument driver.
- Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Note: This function creates a new session each time you invoke it. Although you can open more than one IVI session for the same resource, it is best not to do so. You can use the same session in multiple program threads. You can use the `ciLxs_LockSession` and `ciLxs_UnlockSession` functions to protect sections of code that require exclusive access to the resource.

## Parameter List

resourceName

Variable Type	ViRsrc
---------------	--------

Pass the resource name of the device to initialize.

Refer to the following table below for the exact grammar to use for this parameter. Optional fields are shown in square brackets ([]).

Interface	Syntax
-----------	--------

```
-----
GPIB      GPIB[board]::<primary address>
          [::secondary address]::INSTR
```

```
Serial    ASRL<port>::INSTR
```

Use the GPIB keyword for GPIB instruments. Use the ASRL keyword for serial instruments.

If you do not specify a value for an optional field, the following values are used:

Optional Field	Value
board	0
secondary address	none (31)

The following table contains example valid resource names.

Resource Name	Description
"GPIB::22::INSTR"	GPIB board 0, primary address 22 no secondary address
"GPIB::22::5::INSTR"	GPIB board 0, primary address 22 secondary address 5
"GPIB1::22::5::INSTR"	GPIB board 1, primary address 22 secondary address 5
"ASRL2::INSTR"	COM port 2

Default Value: "GPIB::1::INSTR"

#### IDQuery

Variable Type      ViBoolean

Specify whether you want the instrument driver to perform an ID Query.

Valid Range:

- VI\_TRUE (1) - Perform ID Query (Default Value)
- VI\_FALSE (0) - Skip ID Query

When you set this parameter to VI\_TRUE, the driver verifies that the instrument you initialize is a type that this driver supports.

Circumstances can arise where it is undesirable to send an ID Query command string to the instrument. When you set this parameter to VI\_FALSE, the function initializes the instrument without performing an ID Query.

Notes:

- (1) If ID Query is disable, you should pass DriverSetup string for

your model.

resetDevice

Variable Type            ViBoolean

Specify whether you want the to reset the instrument during the initialization procedure.

Valid Range:

VI\_TRUE (1) - Reset Device (Default Value)

VI\_FALSE (0) - Don't Reset

optionString

Variable Type            ViConstString

You can use this control to set the initial value of certain attributes for the session. The following table lists the attributes and the name you use in this parameter to identify the attribute.

Name	Attribute Defined Constant
RangeCheck	CILXS_ATTR_RANGE_CHECK
QueryInstrStatus	CILXS_ATTR_QUERY_INSTRUMENT_STATUS
Cache	CILXS_ATTR_CACHE
Simulate	CILXS_ATTR_SIMULATE
RecordCoercions	CILXS_ATTR_RECORD_COERCIONS

The format of this string is, "AttributeName=Value" where AttributeName is the name of the attribute and Value is the value to which the attribute will be set. To set multiple attributes, separate their assignments with a comma.

If you pass NULL or an empty string for this parameter and a VISA resource descriptor for the Resource Name parameter, the session uses the default values for the attributes. The default values for the attributes are shown below:

Attribute Name	Default Value
RangeCheck	VI_TRUE
QueryInstrStatus	VI_TRUE
Cache	VI_TRUE
Simulate	VI_FALSE
RecordCoercions	VI_FALSE

If you pass NULL or an empty string for this parameter and a virtual instrument or logical name for the Resource Name parameter, the session uses the values that you configure for virtual instrument or logical name with the IVI Configuration utility.

You can override the values of the attributes by assigning a value explicitly in a string you pass for this parameter. You do not have

to specify all of the attributes and may leave any of them out. If you do not specify one of the attributes, its default value or the value that you configure with the IVI Configuration utility will be used.

The following are the valid values for ViBoolean attributes:

True: 1, TRUE, or VI\_TRUE  
False: 0, False, or VI\_FALSE

Default Value:

"Simulate=0,RangeCheck=1,QueryInstrStatus=1,Cache=1"

Notes:

(1) For the DriverSetup parameter, you can pass the following strings:

Model: X where X is the instrument type 3000iL,4500iL,4801iL

Here is an example of this option string which turns on simulation and emulates the California Instruments 4801iL:

"Simulate=1,DriverSetup=Model:4801iL"

(2) If you enable IDQuery and don't pass DriverSetup driver automatically detect your instrument model.

(3) If you don't pass these parameters and IDQuery is disabled default model is California Instruments 3000iL.

instrumentHandle

Variable Type ViSession (passed by reference)

Returns a ViSession handle that you use to identify the instrument in all subsequent instrument driver function calls.

Notes:

(1) This function creates a new session each time you invoke it. This is useful if you have multiple physical instances of the same type of instrument.

(2) Avoid creating multiple concurrent sessions to the same physical instrument. Although you can create more than one IVI session for the same resource, it is best not to do so. A better approach is to use the same IVI session in multiple execution threads. You can use functions ciLxs\_LockSession and ciLxs\_UnlockSession to protect sections of code that require exclusive access to the resource.

baudRate

Variable Type ViInt32

Specify the baud rate of the serial port.

Baud Rate Ranges:

9600  
 19200  
 38400  
 57600  
 115200

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_InvalidateAllAttributes***

```
ViStatus ciLxs_InvalidateAllAttributes (ViSession instrumentHandle);
```

Purpose

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

Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
WARNINGS:	
None	
ERRORS:	
BFFA4001	Histogram is not enabled.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the

particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviScope Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviScope Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_LockSession***

```
ViStatus ciLxs_LockSession (ViSession instrumentHandle,
                           ViPBoolean callerHasLock);
```

**Purpose**

This function obtains a multithread lock on the instrument session. Before it does so, it waits until all other execution threads have released their locks on the instrument session.

Other threads might have obtained a lock on this session in the following ways:

- The user's application called `ciLxs_LockSession`.
- A call to the instrument driver locked the session.
- A call to the IVI engine locked the session.

After your call to `ciLxs_LockSession` returns successfully, no other threads can access the instrument session until you call `ciLxs_UnlockSession`.

Use `ciLxs_LockSession` and `ciLxs_UnlockSession` around a sequence of calls to instrument driver functions if you require that the instrument retain its settings through the end of the sequence.

You can safely make nested calls to `ciLxs_LockSession` within the same thread. To completely unlock the session, you must balance each call to `ciLxs_LockSession` with a call to `ciLxs_UnlockSession`. If, however, you use the Caller Has Lock parameter in all calls to `ciLxs_LockSession` and `ciLxs_UnlockSession` within a function, the IVI Library locks the session only once within the function regardless of the number of calls you make to `ciLxs_LockSession`. This allows you to call `ciLxs_UnlockSession` just once at the end of the function.

**Parameter List****instrumentHandle**

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value:    None

**callerHasLock**

Variable Type      ViBoolean (passed by reference)

This parameter serves as a convenience. If you do not want to use this parameter, pass `VI_NULL`.

Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to VI\_FALSE. Pass the address of the same local variable to any other calls you make to ciLxs\_LockSession or ciLxs\_UnlockSession in the same function.

The parameter is an input/output parameter. ciLxs\_LockSession and ciLxs\_UnlockSession each inspect the current value and take the following actions:

- If the value is VI\_TRUE, ciLxs\_LockSession does not lock the session again. If the value is VI\_FALSE, ciLxs\_LockSession obtains the lock and sets the value of the parameter to VI\_TRUE.

- If the value is VI\_FALSE, ciLxs\_UnlockSession does not attempt to unlock the session. If the value is VI\_TRUE, ciLxs\_UnlockSession releases the lock and sets the value of the parameter to VI\_FALSE.

Thus, you can, call ciLxs\_UnlockSession at the end of your function without worrying about whether you actually have the lock.

Example:

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
{
    ViStatus error = VI_SUCCESS;
    ViBoolean haveLock = VI_FALSE;

    if (flags & BIT_1)
    {
        viCheckErr( ciLxs_LockSession(vi, &haveLock));
        viCheckErr( TakeAction1(vi));
        if (flags & BIT_2)
        {
            viCheckErr( ciLxs_UnlockSession(vi, &haveLock));
            viCheckErr( TakeAction2(vi));
            viCheckErr( ciLxs_LockSession(vi, &haveLock));
        }
        if (flags & BIT_3)
            viCheckErr( TakeAction3(vi));
    }

    Error:
    /*
       At this point, you cannot really be sure that
       you have the lock. Fortunately, the haveLock
       variable takes care of that for you.
    */
    ciLxs_UnlockSession(vi, &haveLock);
    return error;
}
```

Return Value

Returns the status code of this operation. The status code either

indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_Measure***

```
ViStatus ciLxs_Measure (ViSession instrumentHandle,
                      ViChar _VI_FAR phase[], ViInt32 maxTimeout,
                      ViInt32 measurementType, ViPReal64 measurement);
```

## Purpose

This function takes a single measurement on the phase you specify.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phase

Variable Type      ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

## Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

## maxTimeout

Variable Type      ViInt32

This control sets max. timeout value.

## measurementType

Variable Type            ViInt32

Pass the measurement you want the power supply to take.

Defined Values:

CILXS\_VAL\_MEASURE\_AC\_VOLTAGE\_RMS - ac rms voltage  
 CILXS\_VAL\_MEASURE\_AC\_CURRENT\_RMS - ac rms current  
 CILXS\_VAL\_MEASURE\_CURRENT\_MAXIMUM - peak current  
 CILXS\_VAL\_MEASURE\_CURRENT\_CRESTFACTOR - current crestfactor  
 CILXS\_VAL\_MEASURE\_AC\_POWER - real power  
 CILXS\_VAL\_MEASURE\_AC\_APPARENT\_POWER - apparent power  
 CILXS\_VAL\_MEASURE\_AC\_REACTIVE\_POWER - reactive power  
 CILXS\_VAL\_MEASURE\_AC\_TOTAL\_POWER - total power  
 CILXS\_VAL\_MEASURE\_AC\_POWER\_FACTOR - output power factor  
 CILXS\_VAL\_MEASURE\_NEUTRAL\_AC\_CURRENT\_RMS - neutral ac rms current  
 (3-phase only)  
 CILXS\_VAL\_MEASURE\_FREQUENCY - output frequency  
 CILXS\_VAL\_MEASURE\_PHASE - output phase

Default Value: CILXS\_VAL\_MEASURE\_AC\_VOLTAGE\_RMS

measurement

Variable Type            ViReal64 (passed by reference)

Returns the measured value.

Units: volts (for voltage measurement)  
       amps (for current measurement)  
       watts (for power measurement)  
       voltamperes (for apparent power measurement)  
       voltamperes reactive (for reactive power measurement)  
       hertz (for current frequency)

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

-----

BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code	Types
-----		
3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

***ciLxs\_MeasureArray***

```
ViStatus ciLxs_MeasureArray (ViSession instrumentHandle,
                             ViChar _VI_FAR phase[], ViInt32 maxTime,
                             ViInt32 measurementType, ViInt32 arraySize,
                             ViReal64 _VI_FAR measurement[],
                             ViPInt32 number_of_measurements);
```

**Purpose**

This function takes a harmonic measurement on the phase you specify.

**Parameter List****instrumentHandle**

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

**phase**

Variable Type            ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: "PHASE1"

**Notes:**

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

**maxTime**

Variable Type            ViInt32

This control sets max timeout value.

**measurementType**

Variable Type            ViInt32

Pass the type of measurement to retrieve.

Defined Values:

CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE    CILXS\_VAL\_HARMONIC\_VOLTAGE\_PHASE  
 CILXS\_VAL\_HARMONIC\_CURRENT\_AMPLITUDE    CILXS\_VAL\_HARMONIC\_CURRENT\_PHASE  
 CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_AMPLITUDE  
 CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_PHASE  
 CILXS\_VAL\_MEASURE\_DC\_VOLTAGE  
 CILXS\_VAL\_MEASURE\_DC\_CURRENT  
 CILXS\_VAL\_MEASURE\_NEUTRAL\_DC\_CURRENT

Default Value: CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE

arraySize

Variable Type            ViInt32

Specifies size of measurement array. Harmonic measurements require an array of size 50. All other measurements require an array of size 4096.

measurement

Variable Type            ViReal64[]

Returns the measurements retrieved from the AC Source.

number\_of\_measurements

Variable Type            ViInt32 (passed by reference)

Returns the number of valid values in measurement array.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:

BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code	Types
-----		
3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

***ciLxs\_MeasureHarmonic***

```
ViStatus ciLxs_MeasureHarmonic (ViSession instrumentHandle,
                               ViChar _VI_FAR phase[],
                               ViInt32 maxTimeout, ViInt32 harmonic,
                               ViInt32 measurementType,
                               ViPReal64 measurement);
```

**Purpose**

This function takes a harmonic measurement on the phase you specify.

**Parameter List****instrumentHandle**

Variable Type          ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

**phase**

Variable Type          ViChar[]

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

**Notes:**

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

**maxTimeout**

Variable Type          ViInt32

This control sets max. timeout value.

**harmonic**

Variable Type            ViInt32

Pass the desired harmonic number. Queries sent with a value of 0 return the dc component. A value of 1 returns the fundamental output frequency. Harmonic orders can be queried up to the fundamental measurement bandwidth of the measurement system, which is 12.6kHz. Thus the maximum harmonic that can be measured is dependent on the output frequency. Any harmonics that represent frequencies greater than 12.6kHz are returned as 0.

Default Value: 1

measurementType

Variable Type            ViInt32

Pass the measurement you want the power supply to take.

Defined Values:

- CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE - Voltage Amplitude
- CILXS\_VAL\_HARMONIC\_VOLTAGE\_PHASE     - Voltage Phase
- CILXS\_VAL\_HARMONIC\_CURRENT\_AMPLITUDE - Current Amplitude
- CILXS\_VAL\_HARMONIC\_CURRENT\_PHASE     - Current Phase
- CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_AMPLITUDE - Neutral Current Amplitude
- CILXS\_VAL\_HARMONIC\_NEUTRAL\_CURRENT\_PHASE     - Neutral Current Phase

Default Value: CILXS\_VAL\_HARMONIC\_VOLTAGE\_AMPLITUDE

measurement

Variable Type            ViReal64 (passed by reference)

Returns the measured value.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings

Negative Values                      Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:

BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code	Types
-----		
3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

***ciLxs\_QueryArbWaveformCapabilities***

```

ViStatus ciLxs_QueryArbWaveformCapabilities (ViSession instrumentHandle,
                                             ViPInt32
maximumNumber_ofWaveforms,
                                             ViPInt32 waveformQuantum,
                                             ViPInt32 minimumWaveformSize,
                                             ViPInt32
maximumWaveformSize);

```

## Purpose

This function returns the attributes of the function generator that are related to creating arbitrary sequences. These attributes are the maximum number of sequences, minimum sequence length, maximum sequence length, and maximum loop count.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## maximumNumber\_ofWaveforms

Variable Type      ViInt32 (passed by reference)

Returns the maximum number of arbitrary waveforms that the function generator allows. The driver obtains this value from the `CILXS_ATTR_MAX_NUM_WAVEFORMS` attribute.

## waveformQuantum

Variable Type      ViInt32 (passed by reference)

The size (i.e. number of points) of each waveform must be a multiple of a constant quantum value. This parameter obtains the quantum value the function generator uses. The driver returns this value from the `CILXS_ATTR_WAVEFORM_QUANTUM` attribute. For example, when this attribute returns a value of 8, all waveform sizes must be a multiple of 8.

## minimumWaveformSize

Variable Type      ViInt32 (passed by reference)

Returns the minimum number of points the function generator allows in

a waveform. The driver obtains this value from the CILXS\_ATTR\_MIN\_WAVEFORM\_SIZE attribute.

maximumWaveformSize

Variable Type            ViInt32 (passed by reference)

Returns the maximum number of points the function generator allows in a waveform. The driver obtains this value from the CILXS\_ATTR\_MAX\_WAVEFORM\_SIZE attribute.

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI        Warnings
3FFF0000 to 3FFFFFFF	VISA       Warnings
3FFC0000 to 3FFCFFFF	VXIPnP    Driver Warnings
BFFA0000 to BFFA1FFF	IVI        Errors

BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_QueryDefinedWaveforms***

```
ViStatus ciLxs_QueryDefinedWaveforms (ViSession instrumentHandle,
                                       ViChar _VI_FAR definedWaveforms[]);
```

## Purpose

This function queries for a list of the defined waveform names.

## Notes:

The list includes both pre-defined waveforms such as SINUSOID, SQUARE, and CSINUSOID, as well as any user-defined waveforms.

## Parameter List

## instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## definedWaveforms

Variable Type          ViChar[]

This control displays the list of defined waveform names.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_QueryMaxCurrentLimit***

```
ViStatus ciLxs_QueryMaxCurrentLimit (ViSession instrumentHandle,
                                     ViPReal64 maxCurrentLimit);
```

## Purpose

This function returns the maximum programmable current limit that the power supply accepts for a particular voltage level on a phase for the output range to which the power supply is currently configured.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## maxCurrentLimit

Variable Type            ViReal64 (passed by reference)

This parameter returns the maximum programmable current limit of the AC source.

Units: amps (A)

Note:

1) This value is valid only for sine function.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings

Negative Values                      Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:

BFFA1001    The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code	Types
-----		
3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

***ciLxs\_QueryMaxVoltageLevel***

```
ViStatus ciLxs_QueryMaxVoltageLevel (ViSession instrumentHandle,
                                     ViPReal64 maxVoltageLevel);
```

## Purpose

This function returns the maximum programmable voltage level that the power supply accepts for a particular current limit on a phase for the output range to which the power supply is currently configured.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## maxVoltageLevel

Variable Type            ViReal64 (passed by reference)

This parameter returns the maximum programmable voltage level of the AC source.

Units: volts (V)

Note:

1) This value is valid only for sine function.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings

Negative Values                      Errors

This driver defines the following status codes:

Status	Description
-----	

ERRORS:

BFFA1001    The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code	Types
-----		
3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

***ciLxs\_QueryOutputState***

```
ViStatus ciLxs_QueryOutputState (ViSession instrumentHandle,
                                ViChar _VI_FAR phase[],
                                ViInt32 outputState,
                                ViPBoolean inState);
```

**Purpose**

This function returns whether the power supply is in a particular output state.

An unregulated condition occurs when the output voltage is less than the value of the CILXS\_ATTR\_VOLTAGE\_LEVEL attribute and the current is less than the value of the CILXS\_ATTR\_CURRENT\_LIMIT attribute.

An over-voltage condition occurs when the output voltage is equal to or greater than the value of the CILXS\_ATTR\_OVP\_LIMIT attribute and the CILXS\_ATTR\_OVP\_ENABLED attribute is set to VI\_TRUE.

An over-current condition occurs when the output current is equal to or greater than the value of the CILXS\_ATTR\_CURRENT\_LIMIT attribute and the CILXS\_ATTR\_CURRENT\_LIMIT\_BEHAVIOR attribute is set to CILXS\_VAL\_CURRENT\_TRIP.

When either an over-voltage condition or an over-current condition occurs, the power supply's output protection disables the output. If the power supply is in an over-voltage or over-current state, it does not produce power until the output protection is reset. The `ciLxs_ResetOutputProtection` function resets the output protection. Once the output protection is reset, the power supply resumes generating a power signal.

**Parameter List****instrumentHandle**

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

**phase**

Variable Type	ViChar[]
---------------	----------

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and

swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### outputState

Variable Type            ViInt32

Pass the output state for which you want to query.

Defined Values:

CILXS\_VAL\_OUTPUT\_UNREGULATED            - Unregulated State

CILXS\_VAL\_OUTPUT\_OVER\_VOLTAGE            - Over-voltage State

CILXS\_VAL\_OUTPUT\_OVER\_CURRENT            - Over-current State

CILXS\_VAL\_OUTPUT\_OVER\_TEMPERATURE       - Over-temperature State

Default Value: CILXS\_VAL\_OUTPUT\_UNREGULATED

#### inState

Variable Type            ViBoolean (passed by reference)

This parameter returns VI\_TRUE if the AC Source is currently in the state you specify with the OutputState parameter, and VI\_FALSE if it is not.

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_QueryTrnsListStatus***

```
ViStatus ciLxs_QueryTrnsListStatus (ViSession instrumentHandle,
                                   ViChar _VI_FAR transientStauts[]);
```

## Purpose

This function returns the maximum programmable voltage level that the power supply accepts for a particular current limit on a phase for the output range to which the power supply is currently configured.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## transientStauts

Variable Type            ViChar[]

Returns the transient status string read from the instrument's transient list status queue.

You must pass a ViChar array with at least 256 bytes.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ReadInstrData***

```
ViStatus ciLxs_ReadInstrData (ViSession instrumentHandle,  
                              ViInt32 number_ofBytesToRead,  
                              ViChar _VI_FAR readBuffer[],  
                              ViPInt32 numBytesRead);
```

## Purpose

This function reads data from the instrument.

## Parameter List

## instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## number\_ofBytesToRead

Variable Type          ViInt32

Pass the maximum number of bytes to read from the instruments.

Valid Range: 0 to the number of elements in the Read Buffer.

Default: 0

## readBuffer

Variable Type          ViChar[]

After this function executes, this parameter contains the data that was read from the instrument.

## numBytesRead

Variable Type          ViInt32 (passed by reference)

Returns the number of bytes actually read from the instrument and stored in the Read Buffer.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
WARNINGS:	
none	
ERRORS:	
none	

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviDCPwr Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviDCPwr Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

**ciLxs\_reset**

```
ViStatus ciLxs_reset (ViSession instrumentHandle);
```

Purpose

This function resets the instrument to a known state and sends initialization commands to the instrument. The initialization commands set instrument settings such as Headers Off, Short Command form, and Data Transfer Binary to the state necessary for the operation of the instrument driver.

Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the

different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

## ***ciLxs\_ResetInterchangeCheck***

```
ViStatus ciLxs_ResetInterchangeCheck (ViSession instrumentHandle);
```

### Purpose

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 you execute the test modules 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.

You can use this function to test for such cases. After you call this function, the interchangeability checking algorithms in the specific driver ignore all previous configuration operations. By calling this function at the beginning of a test module, you can determine whether the test module has dependencies on the operation of previously executed test modules.

This function does not clear the interchangeability warnings from the list of previously recorded interchangeability warnings. If you want to guarantee that the `ciLxs_GetNextInterchangeWarning` function only returns those interchangeability warnings that are generated after calling this function, you must clear the list of interchangeability warnings. You can clear the interchangeability warnings list by repeatedly calling the `ciLxs_GetNextInterchangeWarning` function until no more interchangeability warnings are returned. If you are not interested in the content of those warnings, you can call the `ciLxs_ClearInterchangeWarnings` function.

### Parameter List

`instrumentHandle`

Variable Type      `ViSession`

The `ViSession` handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: `None`

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about

the error condition, call the `ciLxs_GetErrorInfo` function. To clear the error information from the driver, call the `ciLxs_ClearErrorInfo` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
WARNINGS:	
none	
ERRORS:	
none	

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviDCPwr Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviDCPwr Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ResetOutputProtection***

```
ViStatus ciLxs_ResetOutputProtection (ViSession instrumentHandle,
                                     ViChar _VI_FAR phaseName[]);
```

## Purpose

This function clears all output-protection conditions on the power supply.

## Parameter List

## instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## phaseName

Variable Type      ViChar[]

Pass the virtual phase name that you assign to the instrument in the Configuration Utility.

Virtual phase names are aliases for instrument-specific phase strings. The instrument-specific phase strings can differ from one instrument to another. Virtual phase names allow you to use and swap instruments without having to change the phase names in your source code. You assign a virtual phase name to an instrument-specific phase through the Configuration Utility. This control accepts virtual phase names you have assigned to the specific instrument you are using. It also accepts the instrument-specific phase names.

Default Value: ""

## Notes:

(1) You can specify the phase name as a string variable or as a literal enclosed in double quotes.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about

the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_ResetWithDefaults***

```
ViStatus ciLxs_ResetWithDefaults (ViSession instrumentHandle);
```

## Purpose

This function resets the instrument and applies initial user specified settings from the Logical Name which was used to initialize the session. If the session was created without a Logical Name, this function is equivalent to the `ciLxs_reset` function.

## Parameter List

instrumentHandle

Variable Type	ViSession
---------------	-----------

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
WARNINGS:	
None	
ERRORS:	
BFFA4001	Histogram is not enabled.

This instrument driver also returns errors and warnings defined by

other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviScope Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviScope Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

### ***ciLxs\_revision\_query***

```
ViStatus ciLxs_revision_query (ViSession instrumentHandle,  
                              ViChar _VI_FAR instrumentDriverRevision[],  
                              ViChar _VI_FAR firmwareRevision[]);
```

#### Purpose

This function returns the revision numbers of the instrument driver and instrument firmware.

#### Parameter List

##### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value:    None

##### instrumentDriverRevision

Variable Type      ViChar[]

Returns the instrument driver software revision numbers in the form of a string.

You must pass a ViChar array with at least 256 bytes.

##### firmwareRevision

Variable Type      ViChar[]

Returns the instrument firmware revision numbers in the form of a string.

You must pass a ViChar array with at least 256 bytes.

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_self\_test***

```
ViStatus ciLxs_self_test (ViSession instrumentHandle,
                        ViPInt16 selfTestResult,
                        ViChar _VI_FAR selfTestMessage[]);
```

## Purpose

This function runs the instrument's self test routine and returns the test result(s).

## Parameter List

## instrumentHandle

Variable Type          ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

## selfTestResult

Variable Type          ViInt16 (passed by reference)

This control contains the value returned from the instrument self test. Zero means success. For any other code, see the device's operator's manual.

Self-Test Code	Description
-----	
0	Passed self test
1	Self test failed

## selfTestMessage

Variable Type          ViChar[]

Returns the self-test response string from the instrument. See the device's operation manual for an explanation of the string's contents.

You must pass a ViChar array with at least 256 bytes.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_SendSoftwareTrigger***

```
ViStatus ciLxs_SendSoftwareTrigger (ViSession instrumentHandle);
```

Purpose

This function sends a command to trigger the power supply. Call this function if you configure the power supply to respond to software triggers. If the power supply is not configured to respond to software triggers, this function returns the error CILXS\_ERROR\_TRIGGER\_NOT\_SOFTWARE.

Notes:

(1) This function is part of the IviDCPwrSoftwareTrigger [SWT] extension group.

Parameter List

instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
-----	-----
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	-----
ERRORS:	

BFFA1001 The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_SetAttributeViBoolean***

```
ViStatus ciLxs_SetAttributeViBoolean (ViSession instrumentHandle,
                                     ViChar _VI_FAR channelName[],
                                     ViAttr attributeID,
                                     ViBoolean attributeValue);
```

**Purpose**

This function sets the value of a ViBoolean attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

**Parameter List**

instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be set. If the attribute is not

channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViBoolean type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViBoolean are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViBoolean

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings

---

3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_SetAttributeViInt32***

```
ViStatus ciLxs_SetAttributeViInt32 (ViSession instrumentHandle,
                                   ViChar _VI_FAR channelName[],
                                   ViAttr attributeID,
                                   ViInt32 attributeValue);
```

**Purpose**

This function sets the value of a ViInt32 attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

**Parameter List**

instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be set. If the attribute is not

channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViInt32 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViInt32 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViInt32

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
-----	

---

3FFA0000 to 3FFA1FFF	IVI	Warnings
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_SetAttributeViReal64***

```
ViStatus ciLxs_SetAttributeViReal64 (ViSession instrumentHandle,
                                     ViChar _VI_FAR channelName[],
                                     ViAttr attributeID,
                                     ViReal64 attributeValue);
```

## Purpose

This function sets the value of a ViReal64 attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

## Parameter List

instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be set. If the attribute is not

channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViReal64 type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViReal64 are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViReal64

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings

---

3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_SetAttributeViSession***

```
ViStatus ciLxs_SetAttributeViSession (ViSession instrumentHandle,
                                     ViChar _VI_FAR channelName[],
                                     ViAttr attributeID,
                                     ViSession attributeValue);
```

**Purpose**

This function sets the value of a ViSession attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

**Parameter List**

instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value:    None

channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be set. If the attribute is not

channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViSession type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViSession are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViSession

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings

---

3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

***ciLxs\_SetAttributeViString***

```
ViStatus ciLxs_SetAttributeViString (ViSession instrumentHandle,
                                     ViChar _VI_FAR channelName[],
                                     ViAttr attributeID,
                                     ViChar _VI_FAR attributeValue[]);
```

## Purpose

This function sets the value of a ViString attribute.

This is a low-level function that you can use to set the values of instrument-specific attributes and inherent IVI attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled and the currently cached value is invalid or is different than the value you specify.

This instrument driver contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

## Parameter List

instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

channelName

Variable Type      ViChar[]

If the attribute is channel-based, this control specifies the name of the channel whose attribute is to be set. If the attribute is not

channel-based, then you set this control to empty string or VI\_NULL.

Pass the virtual channel name that you assign to the instrument in the Configuration Utility.

Virtual channel names are aliases for instrument-specific channel strings. The instrument-specific channel strings can differ from one instrument to another. Virtual channel names allow you to use and swap instruments without having to change the channel names in your source code. You assign a virtual channel name to an instrument-specific channel through the Configuration Utility. This control accepts virtual channel names you have assigned to the specific instrument you are using. It also accepts the instrument-specific channel names.

Default Value: ""

Notes:

(1) You can specify the channel name as a string variable or as a literal enclosed in double quotes.

#### attributeID

Variable Type            ViAttr

Pass the ID of an attribute.

From the function panel window, you can use this control as follows.

- Click on the control or press <ENTER>, <spacebar>, or <ctrl-down arrow>, to display a dialog box containing a hierarchical list of the available attributes. Attributes whose value cannot be set are dim. Help text is shown for each attribute. Select an attribute by double-clicking on it or by selecting it and then pressing <ENTER>.

Read-only attributes appear dim in the list box. If you select a read-only attribute, an error message appears.

A ring control at the top of the dialog box allows you to see all IVI attributes or only the attributes of the ViString type. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. Attributes with data types other than ViString are dim. If you select an attribute data type that is dim, LabWindows/CVI transfers you to the function panel for the corresponding function that is consistent with the data type.

- If you want to enter a variable name, press <CTRL-T> to change this ring control to a manual input box.
- If the attribute in this ring control has named constants as valid values, you can view the constants by moving to the Attribute Value control and pressing <ENTER>.

attributeValue

Variable Type            ViChar[]

Pass the value to which you want to set the attribute.

From the function panel window, you can use this control as follows.

- If the attribute currently showing in the Attribute ID ring control has constants as valid values, you can view a list of the constants by pressing <ENTER> on this control. Select a value by double-clicking on it or by selecting it and then pressing <ENTER>.

Note: Some of the values might not be valid depending on the current settings of the instrument session.

Default Value: none

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI            Warnings

---

3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFA0000 to BFFA1FFF	IVI	Errors
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

---

## ***ciLxs\_StoreRecallRegister***

```
ViStatus ciLxs_StoreRecallRegister (ViSession instrumentHandle,  
                                   ViBoolean command,  
                                   ViInt16 registers);
```

### Purpose

This function saves and recalls up to 8 settings of the AC source (register 0 through 7). The Recall command restores all of the saved states except the trigger system, which is set to the Idle state by an implied Abort Trigger command.

**WARNING:** Recalling a previously stored state may place hazardous voltages at the AC source output.

### Parameter List

#### instrumentHandle

Variable Type      ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

#### command

Variable Type      ViBoolean

Selects Recall or Store command.

0 - for saving register  
1 - for recalling register

#### registers

Variable Type      ViInt16

Selects the register number (0 through 7).

### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_UnlockSession***

```
ViStatus ciLxs_UnlockSession (ViSession instrumentHandle,
                             ViPBoolean callerHasLock);
```

## Purpose

This function releases a lock that you acquired on an instrument session using `ciLxs_LockSession`. Refer to `ciLxs_LockSession` for additional information on session locks.

## Parameter List

`instrumentHandle`

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

`callerHasLock`

Variable Type            ViBoolean (passed by reference)

This parameter serves as a convenience. If you do not want to use this parameter, pass `VI_NULL`.

Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to `VI_FALSE`. Pass the address of the same local variable to any other calls you make to `ciLxs_LockSession` or `ciLxs_UnlockSession` in the same function.

The parameter is an input/output parameter. `ciLxs_LockSession` and `ciLxs_UnlockSession` each inspect the current value and take the following actions:

- If the value is `VI_TRUE`, `ciLxs_LockSession` does not lock the session again. If the value is `VI_FALSE`, `ciLxs_LockSession` obtains the lock and sets the value of the parameter to `VI_TRUE`.
- If the value is `VI_FALSE`, `ciLxs_UnlockSession` does not attempt to unlock the session. If the value is `VI_TRUE`, `ciLxs_UnlockSession` releases the lock and sets the value of the parameter to `VI_FALSE`.

Thus, you can, call `ciLxs_UnlockSession` at the end of your function without worrying about whether you actually have the lock.

## Example:

```
ViStatus TestFunc (ViSession vi, ViInt32 flags)
```

```

{
    ViStatus error = VI_SUCCESS;
    ViBoolean haveLock = VI_FALSE;

    if (flags & BIT_1)
    {
        viCheckErr( ciLxs_LockSession(vi, &haveLock));
        viCheckErr( TakeAction1(vi));
        if (flags & BIT_2)
        {
            viCheckErr( ciLxs_UnlockSession(vi, &haveLock));
            viCheckErr( TakeAction2(vi));
            viCheckErr( ciLxs_LockSession(vi, &haveLock));
        }
        if (flags & BIT_3)
            viCheckErr( TakeAction3(vi));
    }

Error:
    /*
     * At this point, you cannot really be sure that
     * you have the lock. Fortunately, the haveLock
     * variable takes care of that for you.
     */
    ciLxs_UnlockSession(vi, &haveLock);
    return error;
}

```

#### Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by

other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

---

***ciLxs\_WriteArbWaveform***

```
ViStatus ciLxs_WriteArbWaveform (ViSession instrumentHandle,
                                 ViChar _VI_FAR name[],
                                 ViInt32 waveformSize,
                                 ViReal64 _VI_FAR waveformdataArray[]);
```

## Purpose

This function writes the arbitrary waveform to AC power supply.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the `ciLxs_init` or `ciLxs_InitWithOptions` function. The handle identifies a particular instrument session.

Default Value: None

## name

Variable Type            ViChar[]

Specifies the name of user defined waveform.

## Note:

1) User specific function cannot have a name "SINUSOID", "SQUARE", "CSINUSOID". These names are reserved for instrument defined functions.

## waveformSize

Variable Type            ViInt32

Pass the size of the arbitrary waveform you want create.

## Valid Range:

Depends on attributes `CILXS_ATTR_MIN_WAVEFORM_SIZE` and `CILXS_ATTR_MAX_WAVEFORM_SIZE`.

Default Value: 1024

## waveformdataArray

Variable Type            ViReal64[]

Specify the array of data you want to use for the new arbitrary waveform. The array must have at least as many elements as the value you specify in the Waveform Size parameter.

Default Value: None

Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the `ciLxs_error_message` function. To obtain additional information about the error condition, call the `ciLxs_GetError` function. To clear the error information from the driver, call the `ciLxs_ClearError` function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
ERRORS:	
BFFA1001	The trigger source is not software trigger.

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

***ciLxs\_WriteInstrData***

```
ViStatus ciLxs_WriteInstrData (ViSession instrumentHandle,
                              ViChar _VI_FAR writeBuffer[]);
```

## Purpose

This function writes a user-specified string to the instrument.

Note: This function bypasses IVI attribute state caching. Therefore, when you call this function, the cached values for all attributes will be invalidated.

## Parameter List

## instrumentHandle

Variable Type            ViSession

The ViSession handle that you obtain from the ciLxs\_init or ciLxs\_InitWithOptions function. The handle identifies a particular instrument session.

Default Value: None

## writeBuffer

Variable Type            ViChar[]

Pass the string to be written to the instrument.

## Return Value

Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the ciLxs\_error\_message function. To obtain additional information about the error condition, call the ciLxs\_GetError function. To clear the error information from the driver, call the ciLxs\_ClearError function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

This driver defines the following status codes:

Status	Description
-----	

WARNINGS:  
none

ERRORS:  
none

This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code Types
3FFA2000 to 3FFA3FFF	IviDCPwr Warnings
3FFA0000 to 3FFA1FFF	IVI Warnings
3FFF0000 to 3FFFFFFF	VISA Warnings
3FFC0000 to 3FFCFFFF	VXIPnP Driver Warnings
BFFA2000 to BFFA3FFF	IviDCPwr Errors
BFFA0000 to BFFA1FFF	IVI Errors
BFFF0000 to BFFFFFFF	VISA Errors
BFFC0000 to BFFCFFFF	VXIPnP Driver Errors

Attribute Information for the Following Functions:

ciLxs_SetAttributeViInt32	
ciLxs_GetAttributeViInt32	
ciLxs_CheckAttributeViInt32	
ciLxs_SetAttributeViReal64	
ciLxs_GetAttributeViReal64	
ciLxs_CheckAttributeViReal64	
ciLxs_SetAttributeViSession	
ciLxs_GetAttributeViSession	
ciLxs_CheckAttributeViSession	
ciLxs_SetAttributeViBoolean	
ciLxs_GetAttributeViBoolean	
ciLxs_CheckAttributeViBoolean	
ciLxs_SetAttributeViString	
ciLxs_GetAttributeViString	
ciLxs_CheckAttributeViString	
CILXS_ATTR_OUTPUT_PHASE_COUNT	CILXS_ATTR_OUTPUT_PHASE_COUNT
CILXS_ATTR_OUTPUT_ENABLED	CILXS_ATTR_OUTPUT_ENABLED
CILXS_ATTR_OUTPUT_ALC_STATE	CILXS_ATTR_OUTPUT_ALC_STATE
CILXS_ATTR_OUTPUT_PHASE_MODE	CILXS_ATTR_OUTPUT_PHASE_MODE
CILXS_ATTR_SLEW_FREQUENCY_RATE	CILXS_ATTR_SLEW_FREQUENCY_RATE
CILXS_ATTR_FUNCTION	CILXS_ATTR_FUNCTION
CILXS_ATTR_FREQUENCY	CILXS_ATTR_FREQUENCY
CILXS_ATTR_INSTR_USER_TABLE_LIST	
CILXS_ATTR_INSTR_USER_TABLE_LIST	
CILXS_ATTR_CLIPPING_LEVEL	CILXS_ATTR_CLIPPING_LEVEL
CILXS_ATTR_MAX_NUM_WAVEFORMS	CILXS_ATTR_MAX_NUM_WAVEFORMS
CILXS_ATTR_WAVEFORM_QUANTUM	CILXS_ATTR_WAVEFORM_QUANTUM

CILXS_ATTR_MIN_WAVEFORM_SIZE	CILXS_ATTR_MIN_WAVEFORM_SIZE
CILXS_ATTR_MAX_WAVEFORM_SIZE	CILXS_ATTR_MAX_WAVEFORM_SIZE
CILXS_ATTR_TRIGGER_SOURCE	CILXS_ATTR_TRIGGER_SOURCE
CILXS_ATTR_TRIGGER_DELAY	CILXS_ATTR_TRIGGER_DELAY
CILXS_ATTR_TRIGGERED_FREQUENCY_MODE	
CILXS_ATTR_TRIGGERED_FREQUENCY_MODE	
CILXS_ATTR_TRIGGERED_FUNCTION_MODE	
CILXS_ATTR_TRIGGERED_FUNCTION_MODE	
CILXS_ATTR_TRIGGERED_SLEW_FREQUENCY_RATE_MODE	
CILXS_ATTR_TRIGGERED_SLEW_FREQUENCY_RATE_MODE	
CILXS_ATTR_TRIGGERED_FREQUENCY	CILXS_ATTR_TRIGGERED_FREQUENCY
CILXS_ATTR_TRIGGERED_FUNCTION	CILXS_ATTR_TRIGGERED_FUNCTION
CILXS_ATTR_TRIGGERED_SLEW_FREQUENCY_RATE	
CILXS_ATTR_TRIGGERED_SLEW_FREQUENCY_RATE	
CILXS_ATTR_TRIGGER_PULSE_COUNT	CILXS_ATTR_TRIGGER_PULSE_COUNT
CILXS_ATTR_TRIGGER_PULSE_WIDTH	CILXS_ATTR_TRIGGER_PULSE_WIDTH
CILXS_ATTR_TRIGGER_PULSE_PERIOD	
CILXS_ATTR_TRIGGER_PULSE_PERIOD	
CILXS_ATTR_TRIGGER_LIST_COUNT	CILXS_ATTR_TRIGGER_LIST_COUNT
CILXS_ATTR_TRIGGER_LIST_MODE	CILXS_ATTR_TRIGGER_LIST_MODE
CILXS_ATTR_TRIGGER_LIST_MODE	
CILXS_ATTR_ACQUISITION_TRIGGER_SOURCE	
CILXS_ATTR_ACQUISITION_TRIGGER_SOURCE	
CILXS_ATTR_ACQUISITION_START_TIME	
CILXS_ATTR_ACQUISITION_START_TIME	
CILXS_ATTR_ACQUISITION_TIME_INTERVAL	
CILXS_ATTR_ACQUISITION_TIME_INTERVAL	
CILXS_ATTR_OUTPUT_TRIGGER_ENABLED	
CILXS_ATTR_OUTPUT_TRIGGER_ENABLED	
CILXS_ATTR_OUTPUT_TRIGGER_SOURCE	
CILXS_ATTR_OUTPUT_TRIGGER_SOURCE	
CILXS_ATTR_TRIGGER_SYNCHRONIZATION_SOURCE	
CILXS_ATTR_TRIGGER_SYNCHRONIZATION_SOURCE	
CILXS_ATTR_TRIGGER_SYNCHRONIZATION_PHASE	
CILXS_ATTR_TRIGGER_SYNCHRONIZATION_PHASE	
CILXS_ATTR_OUTPUT_PROTECTION_DELAY	
CILXS_ATTR_OUTPUT_PROTECTION_DELAY	
CILXS_ATTR_VOLTAGE_RANGE_MINIMUM	
CILXS_ATTR_VOLTAGE_RANGE_MINIMUM	
CILXS_ATTR_VOLTAGE_RANGE_MAXIMUM	
CILXS_ATTR_VOLTAGE_RANGE_MAXIMUM	
CILXS_ATTR_VOLTAGE_MAXIMUM	CILXS_ATTR_VOLTAGE_MAXIMUM
CILXS_ATTR_CURRENT_MAXIMUM	CILXS_ATTR_CURRENT_MAXIMUM
CILXS_ATTR_FREQUENCY_MINIMUM	CILXS_ATTR_FREQUENCY_MINIMUM
CILXS_ATTR_FREQUENCY_MAXIMUM	CILXS_ATTR_FREQUENCY_MAXIMUM
CILXS_ATTR_ID_QUERY_RESPONSE	CILXS_ATTR_ID_QUERY_RESPONSE

CILXS\_ATTR\_ACQUISITION\_START\_TIME  
 Data Type: ViReal64  
 Description:

CILXS\_ATTR\_ACQUISITION\_TIME\_INTERVAL  
 Data Type: ViReal64  
 Description:

**CILXS\_ATTR\_ACQUISITION\_TRIGGER\_SOURCE**

Data Type: ViInt32

Description:

Values:

CILXS_VAL_TRIG_EXTERNAL	0
CILXS_VAL_SOFTWARE_TRIG	0
CILXS_VAL_TRIG_TTLT	0

**CILXS\_ATTR\_CLIPPING\_LEVEL**

Data Type: ViReal64

Description:

**CILXS\_ATTR\_CURRENT\_MAXIMUM**

Data Type: ViReal64

Description:

**CILXS\_ATTR\_FREQUENCY**

Data Type: ViReal64

Description:

**CILXS\_ATTR\_FREQUENCY\_MAXIMUM**

Data Type: ViReal64

Description:

**CILXS\_ATTR\_FREQUENCY\_MINIMUM**

Data Type: ViReal64

Description:

**CILXS\_ATTR\_FUNCTION**

Data Type: ViString

Description:

**CILXS\_ATTR\_ID\_QUERY\_RESPONSE**

Data Type: ViString

Restrictions: Not settable.

Description:

**CILXS\_ATTR\_INSTR\_USER\_TABLE\_LIST**

Data Type: ViString

Description:

**CILXS\_ATTR\_MAX\_NUM\_WAVEFORMS**

Data Type: ViInt32

Restrictions: Not settable.

Description:

CILXS\_ATTR\_MAX\_WAVEFORM\_SIZE  
 Data Type: ViInt32  
 Restrictions: Not settable.  
 Description:

CILXS\_ATTR\_MIN\_WAVEFORM\_SIZE  
 Data Type: ViInt32  
 Restrictions: Not settable.  
 Description:

CILXS\_ATTR\_OUTPUT\_ALC\_STATE  
 Data Type: ViInt32  
 Description:  
 Values:

CILXS_VAL_ALC_ON	0
CILXS_VAL_ALC_OFF	1
CILXS_VAL_ALC_REGULATE	2

CILXS\_ATTR\_OUTPUT\_ENABLED  
 Data Type: ViBoolean  
 Description:

CILXS\_ATTR\_OUTPUT\_PHASE\_COUNT  
 Data Type: ViInt32  
 Description:  
 Values:

CILXS_VAL_1_PHASE	0
CILXS_VAL_3_PHASE	0

CILXS\_ATTR\_OUTPUT\_PHASE\_MODE  
 Data Type: ViInt32  
 Description:  
 Values:

CILXS_VAL_1_PHASE	0
CILXS_VAL_3_PHASE	0

CILXS\_ATTR\_OUTPUT\_PROTECTION\_DELAY  
 Data Type: ViReal64  
 Description:

CILXS\_ATTR\_OUTPUT\_TRIGGER\_ENABLED  
 Data Type: ViBoolean  
 Description:

## CILXS\_ATTR\_OUTPUT\_TRIGGER\_SOURCE

Data Type: ViInt32

Description:

Values:

CILXS_VAL_OUTPUT_TRIGGER_SOURCE_BOT	0
CILXS_VAL_OUTPUT_TRIGGER_SOURCE_EOT	0
CILXS_VAL_OUTPUT_TRIGGER_SOURCE_LIST	0

## CILXS\_ATTR\_SLEW\_FREQUENCY\_RATE

Data Type: ViReal64

Description:

## CILXS\_ATTR\_TRIGGER\_DELAY

Data Type: ViReal64

Description:

## CILXS\_ATTR\_TRIGGER\_LIST\_COUNT

Data Type: ViInt32

Description:

## CILXS\_ATTR\_TRIGGER\_LIST\_MODE

Data Type: ViInt32

Description:

Values:

CILXS_VAL_TRIGGER_LIST_STEP_ONCE	0
CILXS_VAL_TRIGGER_LIST_STEP_AUTO	0

## CILXS\_ATTR\_TRIGGER\_PULSE\_COUNT

Data Type: ViInt32

Description:

## CILXS\_ATTR\_TRIGGER\_PULSE\_PERIOD

Data Type: ViReal64

Description:

## CILXS\_ATTR\_TRIGGER\_PULSE\_WIDTH

Data Type: ViReal64

Description:

## CILXS\_ATTR\_TRIGGER\_SOURCE

Data Type: ViInt32

Description:

Values:

CILXS_VAL_TRIG_IMMEDIATE	0
CILXS_VAL_TRIG_EXTERNAL	0

```

CILXS_VAL_SOFTWARE_TRIG                                0

CILXS_ATTR_TRIGGER_SYNCHRONIZATION_PHASE
Data Type:      ViReal64
Description:

CILXS_ATTR_TRIGGER_SYNCHRONIZATION_SOURCE
Data Type:      ViInt32
Description:
Values:
  CILXS_VAL_SYNCHRONIZATION_SOURCE_IMMEDIATE          0
  CILXS_VAL_SYNCHRONIZATION_SOURCE_PHASE              0

CILXS_ATTR_TRIGGERED_FREQUENCY
Data Type:      ViReal64
Description:

CILXS_ATTR_TRIGGERED_FREQUENCY_MODE
Data Type:      ViInt32
Description:
Values:
  CILXS_VAL_TRIGGER_MODE_FIX                          0
  CILXS_VAL_TRIGGER_MODE_STEP                         0
  CILXS_VAL_TRIGGER_MODE_PULSE                       0
  CILXS_VAL_TRIGGER_MODE_LIST                        0

CILXS_ATTR_TRIGGERED_FUNCTION
Data Type:      ViString
Description:

CILXS_ATTR_TRIGGERED_FUNCTION_MODE
Data Type:      ViInt32
Description:
Values:
  CILXS_VAL_TRIGGER_MODE_FIX                          0
  CILXS_VAL_TRIGGER_MODE_STEP                         0
  CILXS_VAL_TRIGGER_MODE_PULSE                       0
  CILXS_VAL_TRIGGER_MODE_LIST                        0

CILXS_ATTR_TRIGGERED_SLEW_FREQUENCY_RATE
Data Type:      ViReal64
Description:

```

CILXS\_ATTR\_TRIGGERED\_SLEW\_FREQUENCY\_RATE\_MODE

Data Type: ViInt32

Description:

Values:

CILXS_VAL_TRIGGER_MODE_FIX	0
CILXS_VAL_TRIGGER_MODE_STEP	0
CILXS_VAL_TRIGGER_MODE_PULSE	0
CILXS_VAL_TRIGGER_MODE_LIST	0

CILXS\_ATTR\_VOLTAGE\_MAXIMUM

Data Type: ViReal64

Description:

CILXS\_ATTR\_VOLTAGE\_RANGE\_MAXIMUM

Data Type: ViReal64

Description:

CILXS\_ATTR\_VOLTAGE\_RANGE\_MINIMUM

Data Type: ViReal64

Description:

CILXS\_ATTR\_WAVEFORM\_QUANTUM

Data Type: ViInt32

Restrictions: Not settable.

Description: