Error Macros for IVI

The ivi.h header file contains macros that you can use in your source code to facilitate error handling. These macros require that you have the following declaration at the top of the function in which the macro appears:

ViStatus error = VI_SUCCESS;

The macros also require that you have the following label near the end of the function:

Error:

Some of these macros require access to a ViSession variable named **vi**, which they pass to Ivi_SetErrorInfo in certain cases. The names of these macros all begin with viCheck.

Normally, you use the macros around function calls, but you can also use them around variables or expressions.

The following describes the behavior of each macro.

checkAlloc(pointer)

If pointer is VI_NULL, assign VI_ERROR_ALLOC to the error variable and jump to the Error label.

checkErr(status)

Assign **status** to the error variable. If **status** is positive, coerce the error variable to zero. If **status** is negative, jump to the Error label.

checkWarn(status)

Assign **status** to the error variable. If negative, jump to the Error label.

viCheckAlloc(pointer)

If **pointer** is VI_NULL, assign VI_ERROR_ALLOC to the error variable, call Ivi_SetErrorInfo with VI_ERROR_ALLOC as the primary error code, and jump to the Error label.

viCheckErr(status)

Assign **status** to the error variable. If **status** is positive, coerce the error variable to zero. If **status** is negative, pass it to Ivi_SetErrorInfo and jump to the Error label.

viCheckErrElab(status, elabString)

Assign **status** to the error variable. If **status** is positive, coerce the error variable to zero. If **status** is negative, pass it and **elabString** to Ivi_SetErrorInfo and jump to the Error label.

viCheckParm(status, parameterPosition, parameterName)

Assign **status** to the error variable. If **status** is positive, coerce the error variable to zero. If **status** is negative, do the following:

- Convert **parameterPosition** into one of the VXI*plug&play* error codes for invalid parameters, and pass it as the secondary error code to Ivi_SetErrorInfo. Pass **status** as the primary error code, and pass **parameterName** as the error elaboration.
- Jump to the Error label.

viCheckWarn(status)

Assign **status** to error. If **status** is nonzero, pass it to Ivi_SetErrorInfo. If **status** is negative, jump to the Error label.

Notice that the checkWarn and viCheckWarn macros preserve warnings whereas the other viCheck macros discard them. Also, viCheckWarn calls Ivi_SetErrorInfo on both warnings and errors, whereas the other macros call Ivi_SetErrorInfo only on errors.

When to Use the viCheck Macros

When returning an error or a warning, each user-callable instrument driver function must set the error information for the session and thread. You can do this by explicitly calling Ivi_SetErrorInfo at the end of the function, or you can use the viCheck macros in the function or in the lower-level routines that the function calls.

You can call the viCheck macros only when the following two conditions are true:

- The function in which it appears has a ViSession parameter named **vi** that is an IVI session handle or VI_NULL.
- The first argument you pass to the macro is either a pointer value, in the case of viCheckAlloc, or a status code that is negative if and only if an error occurs. IVI and VISA functions return such status codes.

It is best to use the viCheck macros at the lowest level in your code where these two conditions are true. You can then use the check versions of the macros at higher levels. All IVI engine functions that take the IVI session handle as a parameter call Ivi_SetErrorInfo when they return errors. Thus, you do not have to use the viCheck macros around calls to IVI functions. Nevertheless, it is harmless to make redundant use of the viCheck macros. The viCheck macros call Ivi_SetErrorInfo in such a way that it does not overwrite existing significant error information.

Examples

The following example shows how to handle errors returned by calls to IVI functions.

checkErr(Ivi_SetAttributeViSession (vi, VI_NULL, IVI_ATTR_IO_SESSION,
0, io));

The following example shows how to handle errors that VISA functions return. This method also works for other libraries that return errors as negative values.

viCheckErr(viSetAttribute (io, VI_ATTR_TMO_VALUE, 5000));

The following example shows how to report an error with an elaboration string.

if (triggerCount > 1 || sampleCount > 1)

viCheckErrElab(IVI_ERROR_INVALID_CONFIGURATION,

"Cannot use single point measurement "

" functions when DMM is configured for"

" multi-point.");

The following example shows how to report a parameter error in a usercallable instrument driver function.

viCheckParm(Ivi_SetAttributeViReal64(vi, VI_NULL, HP34401_ATTR_RESOLUTION, 0, resolution), 4, "Resolution");

Error Reporting for IVI

The IVI engine has an extensive mechanism for reporting errors. Almost all functions in the IVI engine return a negative status code if an error occurs and return VI_SUCCESS (0) if the function succeeds. A few functions return positive values to indicate warnings. The Ivi_GetAttributeViString function returns a positive value if the buffer you pass is not large enough to hold the current attribute value. The positive value indicates the size of the buffer you must pass to obtain the complete value.

The IVI engine functions return error and warning values from several sets of status codes. Some status codes are unique to the IVI Library. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range. The <u>status codes</u> topic lists the numeric ranges of the different sets of status codes. It also contains a listing of all the IVI error codes and the most commonly used VISA status codes.

Each IVI session has the following three attributes for reporting error information:

IVI_ATTR_PRIMARY_ERROR IVI_ATTR_SECONDARY_ERROR IVI_ATTR_ERROR_ELABORATION

Each instrument driver defines its own constant name for these attributes, with the instrument prefix replacing IVI in the name.

You can call Ivi_SetErrorInfo to set all three attributes at once. You can call Ivi_ClearErrorInfo to clear all three attributes at once. You can call Ivi_GetErrorInfo to obtain and then clear the values of all three attributes at once. Each instrument driver exports a *Prefix*_ version of each of the Get and Clear functions.

You also can access the attribute values using the following functions:

- Ivi_SetAttributeViInt32 to set primary or secondary error code
- Ivi_SetAttributeViString to set error elaboration string
- Ivi_GetAttributeViInt32 to obtain primary or secondary error code
- Ivi_GetAttributeViString to obtain error elaboration string

The three attributes describe the first error or warning that occurred since the last call to Ivi_GetErrorInfo or Ivi_ClearErrorInfo on the session. The primary error code specifies the primary reason for the error or warning. If no error or warning occurred, the primary error code is VI_SUCCESS (0). The secondary error code is optional and provides additional information about the error or warning condition. A value of 0 indicates no additional information. The error elaboration parameter is a string that can contain further descriptive information about the error or warning condition.

The IVI engine also maintains a primary error code, secondary error code, and error elaboration string for each execution thread. When you call Ivi_SetErrorInfo or Ivi_ClearErrorInfo on a session, the function sets or clears the error information for both the session and the thread. When you pass VI_NULL for the **vi** parameter to Ivi_SetErrorInfo or Ivi_ClearErrorInfo, the function sets or clears only the error information for the thread. This is useful when you do not have a session handle to pass, which occurs when a call to Ivi_SpecificDriverNew fails. To obtain the error information for the thread, you must call Ivi_GetErrorInfo with the **vi** parameter set to VI_NULL.

Normally, it is the responsibility of the user to decide when to clear the error information by calling *Prefix*_GetErrorInfo or *Prefix*_ClearErrorInfo. If an instrument driver calls Ivi_GetErrorInfo, it must restore the error information by calling Ivi_SetErrorInfo, possibly adding a secondary error code or an elaboration string.

Ivi_SetErrorInfo does not overwrite existing significant error information unless you request it to do so. This allows you to make multiple calls to Ivi_SetErrorInfo at different levels in your instrument driver source code without the risk of losing important error information. It also preserves the information about the first error for the user. Refer to the Ivi_SetErrorInfo function description for more information on this mechanism.

IVI Functions

Expand this book to view an alphabetized list of IVI functions.

Ivi_AddAttributeInvalidation

Usage

ViStatus Ivi_AddAttributeInvalidation(ViSession vi, ViAttr attributeID, ViAttr dependentAttributeID, ViBoolean allInstances);

Purpose

This function creates an invalidation dependency relationship between two attributes. When you set the first attribute to a new value, the IVI engine marks the cache value for the second attribute value as invalid. When an attribute cache value is invalid, any attempt to obtain or change the current value of the attribute causes the IVI engine to invoke the read or write callback function for the attribute regardless of the cache value.

Create a dependency relationship if setting the value of one attribute can cause the value of another attribute to change or become out-of-range in the instrument. When this occurs, the cache value of the second attribute no longer reflects the true state of the instrument.

Two-Way Invalidations

Although you can create a two-way invalidation dependency relationship between attributes, it is rarely the correct thing to do. Cases can occur where changing one instrument setting affects another instrument setting, and changing the second instrument setting affects the first. The proper way to handle this situation is to impose a one-way invalidation model in the instrument driver. Identify one attribute as dominant and the other as dependent. Call Ivi_AddAttributeInvalidation to notify the IVI engine that changing the value of the dominant attribute invalidates the dependent attribute. Range check values for the dependent attribute based on the current setting of the dominant attribute. Do not allow the end-user to set the dependent attribute to a value that would cause the instrument to modify the setting of the dominant attribute.

Param	eters
-------	-------

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handl identifies a particular IVI session.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrume driver defines constant names for all c the user-accessible attributes that app to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrume prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the followin constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, th specific driver include file uses the same constant name that appears in t instrument class include file, except th the specific instrument prefix replaces

		the class prefix. For example, the DMI class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name: #define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE
		For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value th is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the followi constant name:
		#define FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BAS + 3L)
		For each attribute that is private to an instrument driver, the instrument drive source file defines a constant name al assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE For example, hp34401a.c defines the following constant name:
		#define HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BA + 1L)
dependentAttributeID	ViAttr	Pass the ID of the attribute to invalidate when the value of the first attribute changes.
allInstances	ViBoolean	Specify whether the invalidation occur on all possible repeated capability instances or only on the instance on which the value of the first attribute

changes. This option is relevant only i both attributes are based on the same repeated capability.

Pass VI_TRUE (1) if you want the invalidation to occur on all repeated capability instances. Otherwise, pass VI_FALSE (0).

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddAttributeViAddr

Usage

ViStatus Ivi_AddAttributeViAddr(ViSession vi, ViAttr newAttributeID, ViChar attributeName[], ViAddr defaultValue, IviAttrFlags flags, ReadAttrViAddr_CallbackPtr readCallback, WriteAttrViAddr_CallbackPtr writeCallback);

Purpose

This function creates and configures a new ViAddr attribute for the instrument session you specify.

You can use ViAddr attributes only for attributes that are private to an instrument driver. You must not make ViAddr attributes accessible to the end-user.



Note Use ViAddr attributes only internally in your driver. Endusers cannot access ViAddr attributes.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that identifies a particular IVI se
newAttributeID	ViAttr	Pass the ID you want the r distinct ID. You must define for the instrument driver or begin with PREFIX_ATTR_
		The include file for your sp names for all of the user-ac includes attributes that the class defines, and attribute Each defined constant nan the specific instrument pre-
		For each IVI engine attribu ivi.h, except replace the IV example, ivi.h defines IVI_ defines the following const
		#define FL45_ATTR_CAC
		For each instrument class appears in the instrument of with the specific instrumen ividmm.h, defines IVIDMM constant name:
		#define FL45_ATTR_RAN
		For each specific instrume constant name in the instru an offset from IVI_SPECIFI defines the following const
		#define FL45_ATTR_HOLD_THR 3L)
		For each attribute that is p

		nam IVI_ follo	SPECIFI	driver source 1 C_PRIVATE_A nstant name:
		#d HI + (efine 234401A 1L)	_ATTR_TRIGG
attributeName	ViChar[]	The cons	name o stant nar	f the attribute. ne for the attri
		For PRE	example FIX_AT	e, if the defined ΓR_RANGE, th
defaultValue	ViAddr	Spe	cify the (default initial va
		The	IVI engi	ne uses the de
			 IVI_A before IVI_V The a before 	TTR_SIMULA] e you set it, and AL_USE_CALI ttribute does n e you set it.
flags	IviAttrFlags	Spe OR neve IVI_	cify the f them tog er cache _VAL_NE	flags you want gether. For exa d, then pass I ^v EVER_CACHE.
		You	can que	ery and modify
		Valio	d Values	
		Bit	Value	Flag
		0	0x0001	IVI_VAL_NO
		1	0x0002	IVI_VAL_NOT
		2	0x0004	IVI_VAL_NOT
		3	0x0008	IVI_VAL_NOT
		4	0x0010	IVI_VAL_NO
		5	0x0020	IVI_VAL_NE\

- 6 0x0040 IVI_VAL_ALV
- 10 0x0400 IVI_VAL_MU
- 11 0x0800 IVI_VAL_COF
- 12 0x1000 IVI_VAL_WA]
- 13 0x2000 IVI_VAL_WA]
- 14 0x4000 IVI_VAL_USE
- 15 0x8000 IVI_VAL_DOI

IVI_VAL_HIDDEN is 0x00: IVI_VAL_NOT_USER_REA Use the IVI_VAL_HIDDEN to access.

IVI_VAL_NOT_SUPPORTE attribute but the specific dr

IVI_VAL_NOT_READABL] drivers can query the value value of the attribute.

IVI_VAL_NOT_WRITABLE drivers can modify the valu the value of the attribute.

IVI_VAL_NOT_USER_REA the value of the attribute. C query the value of the attril

IVI_VAL_NOT_USER_WRI the value of the attribute. C modify the value of the attr

IVI_VAL_NEVER_CACHEvalue of the attribute, regal attribute. The IVI engine al attribute, if present.

IVI_VAL_ALWAYS_CACHI the attribute, if it is valid, re attribute.

IVI_VAL_MULTI_CHANNI

value for each channel. Yo

IVI_VAL_COERCEABLE_(coerces values in a way th software. Do not use this fl undocumented or too com callback. When you query the IVI engine ignores the from the instrument. Thus, engine invokes the read ca function. When you set this allow it to retain most of the

- 1. The instrument alw
- 2. If you send the instrument, the instrument, the instrument, the instrument would be assumptions, the attribute when y value that you prevone or both of these IVI_VAL_NEVER_C

IVI_VAL_WAIT_FOR_OPC the operation complete cal callback for the attribute.

IVI_VAL_WAIT_FOR_OPC the operation complete cal for the attribute.

IVI_VAL_USE_CALLBACH invoke the read and write c mode.

IVI_VAL_DONT_CHECK_ the PREFIX_GetAttribute or driver and the IVI_ATTR_C enabled, the IVI engine cal calling the read or write ca engine never to call the ch

readCallback ReadAttrViAddr_CallbackPtr Specify the read callback f

You must define the read c instrument driver. The func ViStatus You must define the write c instrument driver. The func ViStatus VI FUNC Callbac repCapName, ViAttr attribute Upon entry to the callback, the callback, *value must c N **Note** If you want to box to develop your i parameter names as If you do not want to use a You can change the callba Ivi SetAttrReadCallbackViA writeCallback WriteAttrViAddr CallbackPtr Specify the write callback f you set the attribute to a ne You must define the write c instrument driver. The func

> ViStatus_ VI_FUNC Callbac repCapName, ViAttr attribute

> request the current value c

Note If you want to box to develop your parameter names as

If you do not want to use a

You can change the callba Ivi_SetAttrWriteCallbackViA

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddAttributeViBoolean

Usage

ViStatus Ivi_AddAttributeViBoolean(ViSession vi, ViAttr newAttributeID, ViChar attributeName[], ViBoolean defaultValue, IviAttrFlags flags, ReadAttrViBoolean_CallbackPtr readCallback, WriteAttrViBoolean_CallbackPtr writeCallback);

Purpose

This function creates and configures a new ViBoolean attribute for the instrument session you specify.

Parameters

Name

vi

Туре

I Y

ViSession

newAttributeID ViAttr

Description

The ViSession handle t identifies a particular $\ensuremath{\mathsf{IV}}$

Pass the ID you want tl distinct ID. You must de for the instrument drive begin with PREFIX_AT^{*}.

The include file for you names for all of the use includes attributes that class defines, and attrib Each defined constant the specific instrument

For each IVI engine att ivi.h, except replace the example, ivi.h defines I defines the following cc

#define FL45_ATTR_C

For each instrument cla appears in the instrume with the specific instrum ividmm.h, defines IVID constant name:

#define FL45_ATTR_F

For each specific instru constant name in the ir an offset from IVI_SPE(defines the following cc

#define FL45_ATTR_HOLD_1 3L)

For each attribute that i

		name in the driver sour IVI_SPECIFIC_PRIVAT following constant nam
		#define HP34401A_ATTR_TR + 1L)
attributeName	ViChar[]	The name of the attribu constant name for the a
		For example, if the defi PREFIX_ATTR_RANGE
defaultValue	ViBoolean	Specify the default initia
		The IVI engine uses the
		 IVI_ATTR_SIMU before you set it, IVI_VAL_USE_C The attribute doe
		before you set it.
		Defined Values:
		VI_TRUE (1)VI_FALSE
flags	IviAttrFlags	Specify the flags you w OR them together. For never cached, then pas IVI_VAL_NEVER_CAC
		You can query and mor and Ivi_SetAttributeFlag
		Valid Values:
		Bit Value Flag
		0 0x0001 IVI_VAL_]
		1 0x0002 IVI_VAL_]
		2 0x0004 IVI_VAL_]
		3 0x0008 IVI_VAL_]

- 4 0x0010 IVI_VAL_]
- 5 0x0020 IVI_VAL_]

6 0x0040 IVI_VAL_

- 10 0x0400 IVI_VAL_]
- 11 0x0800 IVI_VAL_(
- 12 0x1000 IVI_VAL_'
- 13 0x2000 IVI_VAL_'
- 14 0x4000 IVI_VAL_1
- 15 0x8000 IVI_VAL_]

IVI_VAL_HIDDEN is 0> IVI_VAL_NOT_USER_I Use the IVI_VAL_HIDI to access.

IVI_VAL_NOT_SUPPOI attribute but the specific

IVI_VAL_NOT_READA drivers can query the value of the attribute.

IVI_VAL_NOT_WRITA drivers can modify the v the value of the attribut

IVI_VAL_NOT_USER_I the value of the attribut query the value of the ϵ

IVI_VAL_NOT_USER_V the value of the attribut modify the value of the

IVI_VAL_NEVER_CAC value of the attribute, re attribute. The IVI enginattribute, if present.

IVI_VAL_ALWAYS_CA the attribute, if it is valic

attribute.

IVI_VAL_MULTI_CHAI value for each channel.

IVI_VAL_COERCEABL coerces values in a wa software. Do not use th undocumented or too c callback. When you que the IVI engine ignores 1 from the instrument. Th engine invokes the read function. When you set allow it to retain most o

- 1. The instrument
- 2. If you send the i instrument, the two assumption the attribute who value that you p one or both of th IVI_VAL_NEVE

IVI_VAL_WAIT_FOR_C the operation complete callback for the attribut

IVI_VAL_WAIT_FOR_C the operation complete for the attribute.

IVI_VAL_USE_CALLB, invoke the read and wr mode.

IVI_VAL_DONT_CHEC the PREFIX_GetAttribut driver and the IVI_ATT enabled, the IVI engine calling the read or write engine never to call the

ReadAttrViBoolean CallbackPtr readCallback Specify the read callba request the current valu You must define the rea instrument driver. The f ViStatus VI FUNC Call repCapName, ViAttr attri Upon entry to the callba the callback, *value mu M Note If you want box to develop yc parameter names If you do not want to us You can change the ca Ivi SetAttrReadCallback writeCallback WriteAttrViBoolean CallbackPtr Specify the write callba you set the attribute to You must define the wr instrument driver. The f ViStatus VI FUNC Call repCapName, ViAttr attri M **Note** If you want box to develop yo parameter names If you do not want to us You can change the ca Ivi_SetAttrWriteCallback

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes
Ivi_AddAttributeViInt32

Usage

ViStatus Ivi_AddAttributeViInt32(ViSession vi, ViAttr newAttributeID, ViChar attributeName[], ViInt32 defaultValue, IviAttrFlags flags, ReadAttrViInt32_CallbackPtr readCallback, WriteAttrViInt32_CallbackPtr writeCallback, IviRangeTablePtr rangeTable);

Purpose

This function creates and configures a new ViInt32 attribute for the instrument session you specify.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that Ivi_SpecificDriverNew. The session.
newAttributeID	ViAttr	Pass the ID you want the r attribute must have a distir name for the attribute in th driver or in your source co with PREFIX_ATTR_, wher prefix.
		The include file for your sp constant names for all of tl apply to the driver. This inc defines, attributes that the attributes that are specific defined constant name be PREFIX is the specific inst
		For each IVI engine attribu appears in ivi.h, except rep instrument prefix. For exar IVI_ATTR_CACHE, and th defines the following conta #define EL45_ATTR_CAC
		For each instrument class name that appears in the i replace the class prefix wit example, the DMM class ii IVIDMM_ATTR_RANGE, & constant name:
		#define FL45_ATTR_RAN
		For each specific instrume access, define a constant include file, and assign a v

		IVI_SPECIFIC_PUBLIC_AT defines the following const #define FL45_ATTR_HOL (IVI_SPECIFIC_PUBLIC_ For each attribute that is p a constant name in the driv that is an offset from IVI_S For example, hp34401a.c d #define HP34401A_ATTR (IVI_SPECIFIC_PRIVATE
attributeName	ViChar[]	The name of the attribute. as the defined constant na
		For example, if the definec PREFIX_ATTR_RANGE, th
defaultValue	Vilnt32	Specify the default initial v
		The IVI engine uses the de
		 IVI_ATTR_SIMULA: attribute value befor IVI_VAL_USE_CAL: attribute is not set. The attribute does n query the attribute b
flags	IviAttrFlags	Specify the flags you want multiple flags, bitwise OR 1 want the attribute to be hic IVI_VAL_NOT_USER_WR
		You can query and modify Ivi_GetAttributeFlags and Iv
		Valid Values:
		Bit Value Flag
		0 0x0001 IVI_VAL_NO
		I UXUUUZ IVI_VAL_NU

- 2 0x0004 IVI_VAL_NO
- 3 0x0008 IVI_VAL_NO
- 4 0x0010 IVI_VAL_NO
- 5 0x0020 IVI_VAL_NEV
- 6 0x0040 IVI_VAL_ALV
- 10 0x0400 IVI_VAL_MU
- 11 0x0800 IVI_VAL_COI
- 12 0x1000 IVI_VAL_WA
- 13 0x2000 IVI_VAL_WA
- 14 0x4000 IVI_VAL_USE
- 15 0x8000 IVI_VAL_DO]

IVI_VAL_HIDDEN is 0x00 IVI_VAL_NOT_USER_REA IVI_VAL_NOT_USER_WR macro for attributes you dc

IVI_VAL_NOT_SUPPORTE creates the attribute but th it.

IVI_VAL_NOT_READABL nor instrument drivers can Only the IVI engine can qu

IVI_VAL_NOT_WRITABLI nor instrument drivers can Only the IVI engine can m

IVI_VAL_NOT_USER_REA cannot query the value of 1 and instrument drivers can

IVI_VAL_NOT_USER_WR cannot modify the value of and instrument drivers can

IVI_VAL_NEVER_CACHEuse the cache value of the

the IVI_ATTR_CACHE attr the read and write callback

IVI_VAL_ALWAYS_CACH cache value of the attribut of the IVI_ATTR_CACHE ¿

IVI_VAL_MULTI_CHANNI a separate value for each using Ivi_SetAttributeFlags.

IVI_VAL_COERCEABLE_(the instrument coerces val driver cannot anticipate in unless the instrument's coor too complicated to enca callback. When you query this flag is set, the IVI engi obtained the cache value f call an Ivi_SetAttribute func read callback the next time function. When you set this assumptions that allow it to caching:

- 1. The instrument alw same way.
- 2. If you send the inst from the instrument the value. Based of engine does not invattribute when you the same value tha from, the instrumer assumption are not IVI_VAL_NEVER_(

IVI_VAL_WAIT_FOR_OPC engine to call the operation before calling the read call

IVI_VAL_WAIT_FOR_OPC

engine to call the operation after calling the write callba IVI_VAL_USE_CALLBACH IVI engine to invoke the re attribute even when in sim IVI_VAL_DONT_CHECK_ user calls one of the PREF PREFIX SetAttribute function IVI_ATTR_QUERY_INSTR enabled, the IVI engine ca session after calling the re This flag directs the IVI en callback for the attribute. readCallback ReadAttrViInt32 CallbackPtr Specify the read callback f invoke when you request t You must define the read c for the specific instrument following prototype: ViStatus VI FUNC Callbac ViConstString repCapName, Upon entry to the callback Upon exit from the callbac current value. M **Note** If you want to Attributes dialog box source code, retain t the prototype for the If you do not want to use a VI_NULL. You can change the callba Ivi SetAttrReadCallbackViIr writeCallback WriteAttrViInt32 CallbackPtr Specify the write callback 1 invoke when you set the a

		for the specific instrument following prototype: ViStatus _VI_FUNC Callbac
		ViConstString repCapName,
		Note If you want to Attributes dialog box source code, retain t the prototype for the
		lf you do not want to use a VI_NULL.
		You can change the callba Ivi_SetAttrWriteCallbackViI
rangeTable	IviRangeTablePtr	Specify the range table tha and coerce values for this IviRangeTable and IviRang want to use a range table,
		If the valid range for this at settings of other attributes range tables. If so, pass V specify a range table callb Ivi_SetAttrRangeTableCallba

You must define the write (

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddAttributeViInt64

Usage

ViStatus Ivi_AddAttributeViInt64(ViSession Vi, ViAttr newAttributeID, ViChar attributeName[], ViInt64 defaultValue, IviAttrFlags flags, ReadAttrViInt64_CallbackPtr readCallback, WriteAttrViInt64_CallbackPtr writeCallback, IviRangeTablePtr rangeTable);

Purpose

This function creates and configures a new ViInt64 attribute for the instrument session you specify.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that Ivi_SpecificDriverNew. The session.
newAttributeID	ViAttr	Pass the ID you want the r attribute must have a distir name for the attribute in th driver or in your source co with PREFIX_ATTR_, wher prefix.
		The include file for your sp constant names for all of tl apply to the driver. This inc defines, attributes that the attributes that are specific defined constant name be PREFIX is the specific inst
		For each IVI engine attribu appears in ivi.h, except rep instrument prefix. For exar IVI_ATTR_CACHE, and th defines the following conta #define EL45_ATTR_CAC
		For each instrument class name that appears in the i replace the class prefix wit example, the DMM class ii IVIDMM_ATTR_RANGE, & constant name:
		#define FL45_ATTR_RAN
		For each specific instrume access, define a constant include file, and assign a v

		IVI_SPECIFIC_PUBLIC_AT defines the following const #define FL45_ATTR_HOL (IVI_SPECIFIC_PUBLIC_ For each attribute that is p a constant name in the driv that is an offset from IVI_S For example, hp34401a.c d #define HP34401A_ATTR (IVI_SPECIFIC_PRIVATE
attributeName	ViChar[]	The name of the attribute. as the defined constant na
		For example, if the definec PREFIX_ATTR_RANGE, th
defaultValue	Vilnt64	Specify the default initial v
		The IVI engine uses the de
		 IVI_ATTR_SIMULA attribute value befor IVI_VAL_USE_CAL attribute is not set. The attribute does n query the attribute b
flags	IviAttrFlags	Specify the flags you want multiple flags, bitwise OR 1 want the attribute to be hic IVI_VAL_NOT_USER_WR
		You can query and modify Ivi_GetAttributeFlags and Iv
		Valid Values:
		Bit Value Flag
		0 0x0001 IVI_VAL_NO
		1 UXUUU2 IVI_VAL_NO

- 2 0x0004 IVI_VAL_NO
- 3 0x0008 IVI_VAL_NO
- 4 0x0010 IVI_VAL_NO
- 5 0x0020 IVI_VAL_NEV
- 6 0x0040 IVI_VAL_ALV
- 10 0x0400 IVI_VAL_MU
- 11 0x0800 IVI_VAL_COI
- 12 0x1000 IVI_VAL_WA
- 13 0x2000 IVI_VAL_WA
- 14 0x4000 IVI_VAL_USE
- 15 0x8000 IVI_VAL_DO]

IVI_VAL_HIDDEN is 0x00 IVI_VAL_NOT_USER_REA IVI_VAL_NOT_USER_WR macro for attributes you dc

IVI_VAL_NOT_SUPPORTE creates the attribute but th it.

IVI_VAL_NOT_READABL nor instrument drivers can Only the IVI engine can qu

IVI_VAL_NOT_WRITABLI nor instrument drivers can Only the IVI engine can m

IVI_VAL_NOT_USER_REA cannot query the value of 1 and instrument drivers can

IVI_VAL_NOT_USER_WR cannot modify the value of and instrument drivers can

IVI_VAL_NEVER_CACHEuse the cache value of the

the IVI_ATTR_CACHE attr the read and write callback

IVI_VAL_ALWAYS_CACH cache value of the attribut of the IVI_ATTR_CACHE ¿

IVI_VAL_MULTI_CHANNI a separate value for each using Ivi_SetAttributeFlags.

IVI_VAL_COERCEABLE_(the instrument coerces val driver cannot anticipate in unless the instrument's coor too complicated to enca callback. When you query this flag is set, the IVI engi obtained the cache value f call an Ivi_SetAttribute func read callback the next time function. When you set this assumptions that allow it to caching:

- 1. The instrument alw same way.
- 2. If you send the inst from the instrument the value. Based of engine does not invattribute when you the same value tha from, the instrumer assumption are not IVI_VAL_NEVER_(

IVI_VAL_WAIT_FOR_OPC engine to call the operation before calling the read call

IVI_VAL_WAIT_FOR_OPC

engine to call the operation after calling the write callba IVI_VAL_USE_CALLBACH IVI engine to invoke the re attribute even when in sim IVI_VAL_DONT_CHECK_ user calls one of the PREF PREFIX SetAttribute function IVI_ATTR_QUERY_INSTR enabled, the IVI engine ca session after calling the re This flag directs the IVI en callback for the attribute. readCallback ReadAttrViInt64 CallbackPtr Specify the read callback f invoke when you request t You must define the read c for the specific instrument following prototype: ViStatus VI FUNC Callbac ViConstString repCapName, Upon entry to the callback Upon exit from the callbac current value. M **Note** If you want to Attributes dialog box source code, retain t the prototype for the If you do not want to use a VI_NULL. You can change the callba Ivi SetAttrReadCallbackViIr writeCallback WriteAttrViInt64 CallbackPtr Specify the write callback 1 invoke when you set the a

		for the specific instrument following prototype: ViStatus _VI_FUNC Callbac
		ViConstString repCapName,
		Note If you want to Attributes dialog box source code, retain t the prototype for the
		lf you do not want to use a VI_NULL.
		You can change the callba Ivi_SetAttrWriteCallbackViI
rangeTable	IviRangeTablePtr	Specify the range table tha and coerce values for this IviRangeTable and IviRang want to use a range table,
		If the valid range for this at settings of other attributes range tables. If so, pass V specify a range table callb Ivi_SetAttrRangeTableCallba

You must define the write (

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddAttributeViReal64

Usage

ViStatus Ivi_AddAttributeViReal64(ViSession vi, ViAttr newAttributeID, ViChar attributeName[], ViReal64 defaultValue, IviAttrFlags flags, ReadAttrViReal64_CallbackPtr readCallback, WriteAttrViReal64_CallbackPtr writeCallback, IviRangeTablePtr rangeTable, ViInt32 comparePrecision);

Purpose

This function creates and configures a new ViReal64 attribute for the instrument session you specify.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handl identifies a particular
newAttributeID	ViAttr	Pass the ID you wan distinct ID. You must for the instrument dri begin with PREFIX_A
		The include file for ye names for all of the u includes attributes th class defines, and at Each defined consta the specific instrume
		For each IVI engine a ivi.h, except replace example, ivi.h define defines the following
		#define FL45_ATTR
		For each instrument appears in the instru with the specific instr ividmm.h, defines IV constant name:
		#define FL45_ATTR
		For each specific ins constant name in the an offset from IVI_SI defines the following #define FL45_ATTR_HOLD
		3L)
		For each attribute the

		name in IVI_SPE following	the (CIFI(co r	driver sc C_PRIV/ Istant na
		#define HP344 + 1L)) 1A_	_ATTR_]
attributeName	ViChar[]	The nam constant	ne of : nan	the attri ne for th
		For exar PREFIX_	nple _ATT	, if the d R_RAN
defaultValue	ViReal64	Specify to The IVI of IV be IV • Th be	the d engir I_AT fore I_VA ne at efore	lefault ir ne uses TR_SIN you set L_USE tribute d you set
flags	IviAttrFlags	Specify to OR then never ca IVI_VAL You can and Ivi_to Valid Val	the fl tog cheo _NE que SetA	ags you ether. F d, then r VER_C ry and n ttributeF
		Bit Valu	Je	Flag
		0 0x0	001	IVI_VAI
		1 0x0	002	IVI_VAI
		2 0x0	004	IVI_VAI
		4 0x0	010	IVI VAI
		5 0x0	020	_ IVI_VAI

- 6 0x0040 IVI_VAI
- 10 0x0400 IVI_VAI

11 0x0800 IVI_VAI

- 12 0x1000 IVI_VAI
- 13 0x2000 IVI_VAI
- 14 0x4000 IVI_VAI
- 15 0x8000 IVI_VAI

IVI_VAL_HIDDEN is IVI_VAL_NOT_USEF Use the IVI_VAL_HI to access.

IVI_VAL_NOT_SUPP attribute but the spec

IVI_VAL_NOT_REAI drivers can query the value of the attribute

IVI_VAL_NOT_WRI drivers can modify th the value of the attrik

IVI_VAL_NOT_USEF the value of the attrik query the value of the

IVI_VAL_NOT_USEF the value of the attrik modify the value of tl

IVI_VAL_NEVER_CA value of the attribute attribute. The IVI enc attribute, if present.

IVI_VAL_ALWAYS_(the attribute, if it is va attribute.

IVI_VAL_MULTI_CH

value for each chanr

IVI_VAL_COERCEA coerces values in a v software. Do not use undocumented or too callback. When you of the IVI engine ignore from the instrument. engine invokes the ro function. When you s allow it to retain mos

- 1. The instrume
- 2. If you send th instrument, th two assumptithe attribute v value that you one or both o IVI_VAL_NEV

IVI_VAL_WAIT_FOR the operation comple callback for the attrib

IVI_VAL_WAIT_FOR the operation comple for the attribute.

IVI_VAL_USE_CALI invoke the read and ' mode.

IVI_VAL_DONT_CHI the PREFIX_GetAttril driver and the IVI_A^r enabled, the IVI engi calling the read or wi engine never to call t

readCallback

ReadAttrViReal64_CallbackPtr S

^{tr} Specify the read call

		reque	est the current v
		You r instru	nust define the Iment driver. Th
		ViSta repCa	tus _VI_FUNC C pName, ViAttr at
		Upon the c	entry to the cal allback, *value r
			Note If you wa box to develop parameter nam
		lf you	ı do not want to
		You o Ivi_S	can change the etAttrReadCallba
writeCallback	WriteAttrViReal64_CallbackPtr	Spec you s	ify the write call set the attribute
		You r instru	nust define the ' Iment driver. Th
		ViSta repCa	tus _VI_FUNC C pName, ViAttr at
		2	Note If you wa box to develop parameter nam
		lf you	l do not want to
		You o Ivi_S	can change the etAttrWriteCallba
rangeTable	lviRangeTablePtr	Spec value IviRa VI_N	ify the range tal es for this attribu ngeTableEntry i ULL.
		If the attrib for th	valid range for utes, you might is parameter, ar

Ivi_SetAttrRangeTable

Specify the degree o function uses for this Ivi_SetAttrCompareCa function, the IVI engi comparing cache val you set the attribute you specify, the IVI e

The IVI engine uses differences between

The value for this pa higher the value, the callback to consider

Valid Range: 0, or 1

If you pass 0, the fur which is 14.

You can modify this v Ivi_SetAttrComparePr

comparePrecision Vilnt32

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddAttributeViSession

Usage

ViStatus Ivi_AddAttributeViSession(ViSession vi, ViAttr newAttributeID, ViChar attributeName[], ViSession defaultValue, IviAttrFlags flags, ReadAttrViSession_CallbackPtr readCallback, WriteAttrViSession_CallbackPtr writeCallback);

Purpose

This function creates and configures a new ViSession attribute for the instrument session you specify.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle t identifies a particular IV
newAttributeID	ViAttr	Pass the ID you want the distinct ID. You must defor the instrument drive begin with PREFIX_ATT
		The include file for your names for all of the use includes attributes that class defines, and attrik Each defined constant the specific instrument
		For each IVI engine attu ivi.h, except replace the example, ivi.h defines I defines the following cc
		#define FL45_ATTR_C
		For each instrument cla appears in the instrume with the specific instrun ividmm.h, defines IVIDI constant name:
		#define FL45_ATTR_R
		For each specific instru constant name in the in an offset from IVI_SPE(defines the following cc
		#define FL45_ATTR_HOLD_T 3L)
		For each attribute that i

		nan IVI_ follo	ne in the _SPECIFI wing col	driver sour C_PRIVATI nstant name
		#c Hi +	lefine P34401A <u></u> 1L)	_ATTR_TR]
attributeName	ViChar[]	The con	e name o stant nar	f the attribu ne for the ε
		For PRE	example EFIX_AT	e, if the defi FR_RANGE
defaultValue	ViSession	Spe	ecify the o	default initia
		The	IVI engi	ne uses the
			 IVI_A' before IVI_V. The a before 	TTR_SIMU e you set it, AL_USE_C. ttribute doe e you set it.
flags	IviAttrFlags	Spe OR nev IVI_	ecify the f them tog er cache _VAL_NE	lags you w gether. For d, then pas EVER_CAC
		You and	can que Ivi_SetA	ry and moc ttributeFlag
		Vali	d Values	:
		Bit	Value	Flag
		0	0x0001	IVI_VAL_I
		1	0x0002	IVI_VAL_I
		2	0x0004	IVI_VAL_I
		3	0x0008	IVI_VAL_I
		4	$(0 \times 0 \times 0)$	$I V I V \Delta I$
		_		

- 6 0x0040 IVI_VAL_/
- 10 0x0400 IVI_VAL_1
- 11 0x0800 IVI_VAL_(
- 12 0x1000 IVI_VAL_V
- 13 0x2000 IVI_VAL_V
- 14 0x4000 IVI_VAL_I
- 15 0x8000 IVI_VAL_I

IVI_VAL_HIDDEN is 0> IVI_VAL_NOT_USER_F Use the IVI_VAL_HIDE to access.

IVI_VAL_NOT_SUPPOF attribute but the specific

IVI_VAL_NOT_READA drivers can query the value of the attribute.

IVI_VAL_NOT_WRITAI drivers can modify the v the value of the attribute

IVI_VAL_NOT_USER_F the value of the attribut query the value of the a

IVI_VAL_NOT_USER_V the value of the attribut modify the value of the

IVI_VAL_NEVER_CAC: value of the attribute, re attribute. The IVI engine attribute, if present.

IVI_VAL_ALWAYS_CA the attribute, if it is valic attribute.

IVI_VAL_MULTI_CHAT

value for each channel.

IVI_VAL_COERCEABL coerces values in a way software. Do not use th undocumented or too c callback. When you que the IVI engine ignores t from the instrument. Th engine invokes the read function. When you set allow it to retain most o

- 1. The instrument a
- 2. If you send the i instrument, the i two assumptions the attribute whe value that you p one or both of th IVI_VAL_NEVE

IVI_VAL_WAIT_FOR_C the operation complete callback for the attribute

IVI_VAL_WAIT_FOR_C the operation complete for the attribute.

IVI_VAL_USE_CALLB# invoke the read and wri mode.

IVI_VAL_DONT_CHEC the PREFIX_GetAttribut driver and the IVI_ATT enabled, the IVI engine calling the read or write engine never to call the

readCallback ReadAttrViSession_CallbackPtr Specify the read callback

		requ You instru ViSta repC Upon the c	est the current valu must define the rea ument driver. The f atus _VI_FUNC Call apName, ViAttr attril n entry to the callba callback. *value mu
			Note If you want box to develop yc parameter names
		If yo	u do not want to us
		You Ivi_S	can change the cal SetAttrReadCallback'
writeCallback	WriteAttrViSession_CallbackPtr	Spec you :	cify the write callba set the attribute to a
		You instr	must define the wri ument driver. The f
		ViSta repC	atus _VI_FUNC Call apName, ViAttr attri]
		2	Note If you want box to develop yc parameter names
		If yo	u do not want to us
		You Ivi_S	can change the cal SetAttrWriteCallback

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddAttributeViString

Usage

ViStatus Ivi_AddAttributeViString(ViSession vi, ViAttr newAttributeID, ViChar attributeName[], ViChar defaultValue[], IviAttrFlags flags, ReadAttrViString_CallbackPtr readCallback, WriteAttrViString_CallbackPtr writeCallback);
Purpose

This function creates and configures a new ViString attribute for the instrument session you specify.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tha identifies a particular IVI s
newAttributeID	ViAttr	Pass the ID you want the distinct ID. You must defir for the instrument driver c begin with PREFIX_ATTR
		The include file for your s names for all of the user-a includes attributes that the class defines, and attribut Each defined constant na the specific instrument pre-
		For each IVI engine attrib ivi.h, except replace the l' example, ivi.h defines IVI defines the following cons
		#define FL45_ATTR_CA
		For each instrument class appears in the instrument with the specific instrume ividmm.h, defines IVIDMN constant name:
		#define FL45_ATTR_RA
		For each specific instrum constant name in the inst an offset from IVI_SPECII defines the following cons
		#define FL45_ATTR_HOLD_TH 3L)
		For each attribute that is I

		nam IVI_ follo #d HI	ne in the _SPECIFI owing cor lefine P34401A_	driver s C_PRIN nstant n _ATTR_	ource /ATE_ ame: _TRIG
		+	1L)		
attributeName	ViChar[]	The con	name of stant nar	f the att ne for t	ribute he attı
		For PRE	example CFIX_ATT	e, if the CR_RAN	define NGE, t
defaultValue	ViChar[]	Spe	cify the a	default i	nitial
		The	IVI engi	ne uses	s the c
			 IVI_A' before IVI_V. The a before 	TTR_SI e you se AL_USI ttribute e you se	MULA et it, ar E_CAI does et it.
flags	IviAttrFlags	Spe OR nev IVI_ You and	cify the f them tog er cache _VAL_NE can que Ivi_SetA	lags yo gether. I d, then EVER_C ry and .ttributel	u wan For ex pass CACHI modify Flags.
		Vali	d Values	:	
		Bit	Value	Flag	
		0	0x0001	IVI_VA	L_NC
		1	0x0002	IVI_VA	L_NC
		2	0x0004	IVI_VA	AL_NC
		3	0x0008	IVI_VA	AL_NC
		4	0x0010	IVI_VA	AL_NC
		5	0x0020	IVI_VA	L_NE

- 6 0x0040 IVI_VAL_AL
- 10 0x0400 IVI_VAL_MI
- 11 0x0800 IVI_VAL_CC
- 12 0x1000 IVI_VAL_WA
- 13 0x2000 IVI_VAL_WA
- 14 0x4000 IVI_VAL_US
- 15 0x8000 IVI_VAL_DC

IVI_VAL_HIDDEN is 0x0(IVI_VAL_NOT_USER_RE Use the IVI_VAL_HIDDE to access.

IVI_VAL_NOT_SUPPORT attribute but the specific d

IVI_VAL_NOT_READABI drivers can query the valu value of the attribute.

IVI_VAL_NOT_WRITABL drivers can modify the val the value of the attribute.

IVI_VAL_NOT_USER_RE the value of the attribute. query the value of the attr

IVI_VAL_NOT_USER_WF the value of the attribute. modify the value of the attribute attribute.

IVI_VAL_NEVER_CACHI value of the attribute, rega attribute. The IVI engine a attribute, if present.

IVI_VAL_ALWAYS_CACH the attribute, if it is valid, r attribute.

IVI_VAL_MULTI_CHANN

value for each channel. Y

IVI_VAL_COERCEABLE_ coerces values in a way t software. Do not use this undocumented or too con callback. When you query the IVI engine ignores the from the instrument. Thus engine invokes the read c function. When you set th allow it to retain most of tl

- 1. The instrument alv
- 2. If you send the ins instrument, the ins two assumptions, the attribute when value that you pre one or both of the: IVI_VAL_NEVER_

IVI_VAL_WAIT_FOR_OP(the operation complete ca callback for the attribute.

IVI_VAL_WAIT_FOR_OP(the operation complete ca for the attribute.

IVI_VAL_USE_CALLBAC invoke the read and write mode.

IVI_VAL_DONT_CHECK_ the PREFIX_GetAttribute (driver and the IVI_ATTR_ enabled, the IVI engine ca calling the read or write ca engine never to call the c

readCallback ReadAttrViString_CallbackPtr Specify the read callback

request the current value You must define the read instrument driver. The fun

ViStatus _VI_FUNC Callba repCapName, ViAttr attribu

Unlike the read callback f the current value to the ca the current value by pass function.

Note If you want to box to develop your parameter names a

If you do not want to use

You can change the callback Ivi_SetAttrReadCallbackVi

writeCallback WriteAttrViString_CallbackPtr Specify the write callback you set the attribute to a r

You must define the write instrument driver. The fun

ViStatus _VI_FUNC Callba repCapName, ViAttr attribu

Note If you want to box to develop your parameter names a

If you do not want to use

You can change the callbackVi Ivi_SetAttrWriteCallbackVi

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddRepeatedAttributeViAddr

Usage

ViStatus Ivi_AddRepeatedAttributeViAddr(ViSession vi, ViChar repeatedCapabilityName[], ViAttr newAttributeID, ViChar attributeName[], ViAddr defaultValue, IviAttrFlags flags, ReadAttrViAddr_CallbackPtr readCallback, WriteAttrViAddr_CallbackPtr writeCallback);

Purpose

This function creates and configures a new ViAddr attribute for the instrument session you specify. The value of the Repeated Capability parameter determines the repeated capability to which the attribute applies.

You can use ViAddr attributes only for attributes that are private to an instrument driver. You must not make ViAddr attribute accessible to the end-user.



Note Use ViAddr attributes only internally in your driver. Endusers cannot access ViAddr attributes.

Parameters

Name vi **Type** ViSession

repeatedCapabilityName ViChar[]

newAttributeID

ViAttr

Description

The ViSession h identifies a partic

Pass a string co operate. For inst pass in the string

Pass the ID you distinct ID. You r for the instrumer begin with PREF

The include file 1 names for all of includes attribute class defines, ar Each defined co the specific instr

For each IVI enç ivi.h, except repl example, ivi.h de defines the follov

#define FL45_A

For each instrun appears in the ir with the specific ividmm.h, define constant name:

#define FL45_A

For each specific constant name if an offset from IV defines the follov

		#define FL45_ATTR_H 3L) For each attribut
		name in the driv IVI_SPECIFIC_P following consta
		#define HP34401A_AT [*] + 1L)
attributeName	ViChar[]	The name of the constant name f
		For example, if t PREFIX_ATTR_I
defaultValue	ViAddr	Specify the defa
		The IVI engine $\boldsymbol{\iota}$
		 IVI_ATTR before you IVI_VAL_
		 The attribution before you
flags	IviAttrFlags	Specify the flags OR them togeth never cached, th IVI_VAL_NEVE
		You can query a and Ivi_SetAttrib
		Valid Values:
		Bit Value Fla
		0 0x0001 IVI
		1 0x0002 IVI
		2 0x0004 IVI

- 3 0x0008 IVI
- 4 0x0010 IVI
- 5 0x0020 IVI
- 6 0x0040 IVI
- 10 0x0400 IVI
- 11 0x0800 IVI
- 12 0x1000 IVI
- 13 0x2000 IVI
- 14 0x4000 IVI
- 15 0x8000 IVI

IVI_VAL_HIDDE IVI_VAL_NOT_U Use the IVI_VAI to access.

IVI_VAL_NOT_5 attribute but the

IVI_VAL_NOT_F drivers can quer value of the attri

IVI_VAL_NOT_V drivers can mod the value of the

IVI_VAL_NOT_I the value of the query the value

IVI_VAL_NOT_I the value of the modify the value

IVI_VAL_NEVEI value of the attri attribute. The IV attribute, if prese

IVI_VAL_ALWA` the attribute, if it attribute.

IVI_VAL_MULT: value for each cl

IVI_VAL_COER(coerces values i software. Do not undocumented c callback. When y the IVI engine ig from the instrum engine invokes t function. When y allow it to retain

- 1. The instr
- 2. If you ser instrumer two assu the attrib value tha one or bc IVI_VAL

IVI_VAL_WAIT_ the operation co callback for the *i*

IVI_VAL_WAIT_ the operation co for the attribute.

IVI_VAL_USE_C invoke the read mode.

IVI_VAL_DONT the PREFIX_Get driver and the IV enabled, the IVI

		calling engin	g the read (e never to
readCallback	ReadAttrViAddr_CallbackPtr	Speci reque	ify the read est the curre
		You n instru	nust define ment drive
		ViStat repCa	us _VI_FUI pName, ViA
		Upon the ca	entry to the allback, *va
			Note If yc box to dev parameter
		lf you	do not wai
		You c Ivi_Se	an change etAttrReadC
writeCallback	WriteAttrViAddr_CallbackPtr	Speci you s	ify the write et the attrik
		You n instru	nust define ment drive
		ViStat repCa	us _VI_FUI pName, ViA
			Note If yc box to dev parameter
		lf you	do not wai
		You c Ivi_Se	an change etAttrWriteC

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddRepeatedAttributeViBoolean

Usage

ViStatus Ivi_AddRepeatedAttributeViBoolean(ViSession vi, ViChar repeatedCapabilityName[], ViAttr newAttributeID, ViChar attributeName[], ViBoolean defaultValue, IviAttrFlags flags, ReadAttrViBoolean_CallbackPtr readcallback, WriteAttrViBoolean_CallbackPtr writeCallback);

Purpose

This function creates and configures a new ViBoolean attribute for the instrument session you specify. The value of the Repeated Capability parameter determines the repeated capability to which the attribute applies.

Parameters

Name

vi

Type ViSession

repeatedCapabilityName ViChar[]

newAttributeID

ViAttr

Description

The ViSessic identifies a p

Pass a string operate. For pass in the s

Pass the ID y distinct ID. Y for the instru begin with PI The include 1 names for all includes attri class defines Each definec

For each IVI ivi.h, except example, ivi. defines the fo

the specific i

#define FL4

For each inst appears in th with the spec ividmm.h, de constant nan

#define FL4

For each spe constant nan an offset fror defines the fo

		#define FL45_ATTH 3L)
		For each attr name in the (IVI_SPECIFI(following cor
		#define HP34401A_ + 1L)
attributeName	ViChar[]	The name of constant nan
		For example PREFIX_ATT
defaultValue	ViBoolean	Specify the c
		The IVI engir
		 IVI_A] before IVI_V/ The at before
		Defined Valu
		VI_TRUE (1)
flags	IviAttrFlags	Specify the fl OR them tog never cached IVI_VAL_NE
		You can que and Ivi_SetA
		Valid Values:
		Bit Value

- 0 0x0001
- 1 0x0002
- 2 0x0004
- 3 0x0008
- 4 0x0010
- 5 0x0020
- 6 0x0040
- 10 0x0400
- 11 0x0800
- 12 0x1000
- 13 0x2000
- 14 0x4000
- 15 0x8000

IVI_VAL_HII IVI_VAL_NO Use the IVI_ to access.

IVI_VAL_NO attribute but

IVI_VAL_NO drivers can q value of the ¿

IVI_VAL_NC drivers can n the value of t

IVI_VAL_NO the value of t query the val

IVI_VAL_NO the value of t modify the va

IVI_VAL_NE

value of the a attribute. The attribute, if p IVI_VAL_AL the attribute, attribute. IVI_VAL_MU value for eac IVI_VAL_CO coerces valu software. Do undocument callback. Wh the IVI engin from the inst engine invok function. Wh allow it to ret 1. The ir 2. If you instru two a the at value one o IVI_V IVI_VAL_WA the operatior callback for t IVI_VAL_WA the operatior for the attribu IVI_VAL_US invoke the re mode. IVI_VAL_DC

		the PREFIX_ driver and the enabled, the calling the re engine neve	
readcallback	ReadAttrViBoolean_CallbackPtr	Specify the r request the	
		You must de instrument d	:1 1
		ViStatus _VI_ repCapName,	-
		Upon entry t the callback,	(
		Note box to param	
		If you do not	
		You can cha Ivi_SetAttrRe	j I
writeCallback	WriteAttrViBoolean_CallbackPtr	Specify the vyou set the a	/ a
		You must de instrument d	!
		ViStatus _VI_ repCapName,	-
		Note box to param	
		If you do not	
		You can cha Ivi_SetAttrW	l

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddRepeatedAttributeViInt32

Usage

ViStatus Ivi_AddRepeatedAttributeViInt32(ViSession vi, ViChar repeatedCapabilityName[], ViAttr newAttributeID, ViChar attributeName[], ViInt32 defaultValue, IviAttrFlags flags, ReadAttrViInt32_CallbackPtr readCallback, WriteAttrViInt32_CallbackPtr writeCallback, IviRangeTablePtr rangeTable);

Purpose

This function creates and configures a new ViInt32 attribute for the instrument session you specify. The value of the Repeated Capability parameter determines the repeated capability to which the attribute applies.

Parameters

Name vi **Type** ViSession

repeatedCapabilityName ViChar[]

newAttributeID

ViAttr

Description

The ViSession h identifies a parti

Pass a string co operate. For ins pass in the strin

Pass the ID you distinct ID. You I for the instrume begin with PREF

The include file names for all of includes attribut class defines, au Each defined co the specific instr

For each IVI enq ivi.h, except rep example, ivi.h d defines the follo

#define FL45_/

For each instrun appears in the ir with the specific ividmm.h, define constant name:

#define FL45_/

For each specifi constant name i an offset from IN defines the follo

		#define FL45_ATTR_H 3L)
		For each attribution name in the driv IVI_SPECIFIC_F following consta
		#define HP34401A_AT + 1L)
attributeName	ViChar[]	The name of the constant name f
		For example, if t PREFIX_ATTR_3
defaultValue	Vilnt32	Specify the defa
		The IVI engine ι
		 IVI_ATTR before yo IVI_VAL_
		The attrib before you
flags	IviAttrFlags	Specify the flags OR them togeth never cached, th IVI_VAL_NEVE
		You can query a and Ivi_SetAttrib
		Valid Values:
		Bit Value Fla
		0 0x0001 IVI
		1 0x0002 IVI
		2 0x0004 IVI

- 3 0x0008 IVI
- 4 0x0010 IVI
- 5 0x0020 IVI
- 6 0x0040 IVI
- 10 0x0400 IVI
- 11 0x0800 IVI
- 12 0x1000 IVI
- 13 0x2000 IVI
- 14 0x4000 IVI
- 15 0x8000 IVI

IVI_VAL_HIDDI IVI_VAL_NOT_U Use the IVI_VAI to access.

IVI_VAL_NOT_S attribute but the

IVI_VAL_NOT_I drivers can quer value of the attri

IVI_VAL_NOT_' drivers can mod the value of the

IVI_VAL_NOT_I the value of the query the value

IVI_VAL_NOT_I the value of the modify the value

IVI_VAL_NEVE: value of the attri attribute. The IV attribute, if prese

IVI_VAL_ALWA the attribute, if it attribute.

IVI_VAL_MULT value for each c

IVI_VAL_COER coerces values i software. Do no undocumented (callback. When the IVI engine ig from the instrum engine invokes 1 function. When y allow it to retain

- 1. The instr
- 2. If you se instrume two assu the attrib value tha one or bo IVI_VAL

IVI_VAL_WAIT_ the operation co callback for the

IVI_VAL_WAIT_ the operation co for the attribute.

IVI_VAL_USE_(invoke the read mode.

IVI_VAL_DONT the PREFIX_Get driver and the IV enabled, the IVI

		calling the read engine never to
readCallback	ReadAttrViInt32_CallbackPtr	Specify the read request the curre
		You must define instrument drive
		ViStatus _VI_FU repCapName, Vi/
		Upon entry to th the callback, *va
		Note If yc box to dev parameter
		If you do not wa
		You can change Ivi_SetAttrReadC
writeCallback	WriteAttrViInt32_CallbackPtr	Specify the write you set the attrik
		You must define instrument drive
		ViStatus _VI_FU repCapName, Vi/
		Note If yc box to dev parameter
		If you do not wa
		You can change Ivi_SetAttrWrite(
rangeTable	IviRangeTablePtr	Specify the rang values for this a lviRangeTableE

IVI engine autor range table is IV default coerce c

If you do not wa

If the valid range attributes, you n for this paramete Ivi_SetAttrRange

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddRepeatedAttributeViInt64

Usage

ViStatus Ivi_AddRepeatedAttributeViInt64(ViSession vi, ViChar repeatedCapabilityName[], ViAttr newAttributeID, ViChar attributeName[], ViInt64 defaultValue, IviAttrFlags flags, ReadAttrViInt64_CallbackPtr readCallback, WriteAttrViInt64_CallbackPtr writeCallback, IviRangeTablePtr rangeTable);

Purpose

This function creates and configures a new ViInt64 attribute for the instrument session you specify. The value of the Repeated Capability parameter determines the repeated capability to which the attribute applies.

Parameters

Name vi **Type** ViSession

repeatedCapabilityName ViChar[]

newAttributeID

ViAttr

Description

The ViSession h identifies a parti

Pass a string co operate. For ins pass in the strin

Pass the ID you distinct ID. You I for the instrume begin with PREF

The include file names for all of includes attribut class defines, au Each defined co the specific instr

For each IVI enq ivi.h, except rep example, ivi.h d defines the follo

#define FL45_/

For each instrun appears in the ir with the specific ividmm.h, define constant name:

#define FL45_/

For each specifi constant name i an offset from IN defines the follo

		#define FL45_ATTR_H 3L)
		name in the driv IVI_SPECIFIC_F following consta
		#define HP34401A_AT + 1L)
attributeName	ViChar[]	The name of the constant name f
		For example, if t PREFIX_ATTR_1
defaultValue	Vilnt64	Specify the defa
		The IVI engine ι
		IVI_ATTR before yo
		• The attrib
		before yo
flags	IviAttrFlags	Specify the flage OR them togeth never cached, th IVI_VAL_NEVE
		You can query a and Ivi_SetAttrit
		Valid Values:
		Bit Value Fla
		0 0x0001 IVI
		1 0x0002 IVI
		2 0x0004 IVI
- 3 0x0008 IVI
- 4 0x0010 IVI
- 5 0x0020 IVI
- 6 0x0040 IVI
- 10 0x0400 IVI
- 11 0x0800 IVI
- 12 0x1000 IVI
- 13 0x2000 IVI
- 14 0x4000 IVI
- 15 0x8000 IVI

IVI_VAL_HIDDI IVI_VAL_NOT_I Use the IVI_VAI to access.

IVI_VAL_NOT_S attribute but the

IVI_VAL_NOT_I drivers can quer value of the attri

IVI_VAL_NOT_' drivers can mod the value of the

IVI_VAL_NOT_I the value of the query the value

IVI_VAL_NOT_I the value of the modify the value

IVI_VAL_NEVE: value of the attri attribute. The IV attribute, if prese

IVI_VAL_ALWA the attribute, if it attribute.

IVI_VAL_MULT value for each c

IVI_VAL_COER coerces values i software. Do no undocumented (callback. When the IVI engine ig from the instrum engine invokes i function. When y allow it to retain

- 1. The instr
- 2. If you se instrume two assu the attrib value tha one or bo IVI_VAL

IVI_VAL_WAIT_ the operation co callback for the

IVI_VAL_WAIT_ the operation co for the attribute.

IVI_VAL_USE_(invoke the read mode.

IVI_VAL_DONT the PREFIX_Get driver and the IV enabled, the IVI

		calling the read engine never to
readCallback	ReadAttrViInt64_CallbackPtr	Specify the read request the curre
		You must define instrument drive
		ViStatus _VI_FU repCapName, Vi/
		Upon entry to th the callback, *va
		Note If yc box to dev parameter
		If you do not wa
		You can change Ivi_SetAttrReadC
writeCallback	WriteAttrViInt64_CallbackPtr	Specify the write you set the attrik
		You must define instrument drive
		ViStatus _VI_FU repCapName, Vi/
		Note If yc box to dev parameter
		If you do not wa
		You can change Ivi_SetAttrWrite(
rangeTable	IviRangeTablePtr	Specify the rang values for this a lviRangeTableE

IVI engine autor range table is IV default coerce c

If you do not wa

If the valid range attributes, you n for this paramete Ivi_SetAttrRange

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddRepeatedAttributeViReal64

Usage

ViStatus Ivi_AddRepeatedAttributeViReal64(ViSession vi, ViChar repeatedCapabilityName[], ViAttr newAttributeID, ViChar attributeName[], ViReal64 defaultValue, IviAttrFlags flags, ReadAttrViReal64_CallbackPtr readCallback, WriteAttrViReal64_CallbackPtr writeCallback, IviRangeTablePtr rangeTable, ViInt32 comparePrecision);

Purpose

This function creates and configures a new ViReal64 attribute for the instrument session you specify. The value of the Repeated Capability parameter determines the repeated capability to which the attribute applies.

Parameters

Name vi **Type** ViSession

repeatedCapabilityName ViChar[]

newAttributeID

ViAttr

Description

The ViSessior identifies a pa

Pass a string operate. For in pass in the str

Pass the ID yo distinct ID. Yo for the instrum begin with PR

The include fil names for all includes attrib class defines, Each defined the specific in

For each IVI ϵ ivi.h, except re example, ivi.h defines the fol

#define FL45

For each instr appears in the with the speci ividmm.h, defi constant name

#define FL45

For each spec constant name an offset from defines the fol

		#d FI 3L	efine .45_ATT .)	'R_
		For nam IVI_ follo	each att ne in the SPECIF	ril d IC ns
		#d HI +	efine 234401A 1L)	<i>I</i> -
attributeName	ViChar[]	The con:	name o stant nai	if t mi
		For PRE	example FIX_AT	રે, TF
defaultValue	ViReal64	Spe	cify the	de
		The	IVI engi	in∈
			 IVI_A before IVI_V The a before 	T Ə Y A I ttr Ə Y
flags	IviAttrFlags	Spe OR neve IVI_	cify the t them tog er cache _VAL_NH	fla ge ed EV
		You and	can que Ivi_SetA	ery Att
		Vali	d Values	;;
		Bit	Value	F
		0	0x0001	Ľ
		1	0x0002	ľ
		2	0x0004	I

- 3 0x0008 I
- 4 0x0010 I
- 5 0x0020 I
- 6 0x0040 I
- 10 0x0400 I
- 11 0x0800 I
- 12 0x1000 I
- 13 0x2000 I
- 14 0x4000 I
- 15 0x8000 I

IVI_VAL_HID IVI_VAL_NOT Use the IVI_V to access.

IVI_VAL_NOT attribute but th

IVI_VAL_NOT drivers can qu value of the at

IVI_VAL_NOT drivers can me the value of th

IVI_VAL_NOT the value of th query the valu

IVI_VAL_NOT the value of th modify the val

IVI_VAL_NEV value of the at attribute. The attribute, if pre

IVI_VAL_ALV the attribute, i attribute. IVI_VAL_MUI value for each IVI_VAL_COF coerces value software. Do r undocumente callback. Whe the IVI engine from the instru engine invoke function. Whe allow it to reta 1. The in: 2. If you s instrun two as the att value t one or IVI_VA IVI_VAL_WAI the operation callback for th IVI_VAL_WAI the operation for the attribut IVI_VAL_USE invoke the rea mode. IVI_VAL_DON the PREFIX C driver and the enabled, the I

		calling the rea engine never
readCallback	ReadAttrViReal64_CallbackPtr	Specify the re request the cu
		You must defining instrument drive
		ViStatus _VI_F repCapName, \
		Upon entry to the callback, *
		Note If box to d paramet
		If you do not v
		You can chang Ivi_SetAttrRea
writeCallback	WriteAttrViReal64_CallbackPtr	Specify the wi you set the at
		You must defining instrument drive
		ViStatus _VI_F repCapName, \
		Note If box to d paramet
		If you do not v
		You can chan Ivi_SetAttrWri
rangeTable	IviRangeTablePtr	Specify the ra values for this IviRangeTable

IVI engine aut range table is default coerce If you do not v If the valid rar attributes, you for this param Ivi_SetAttrRan comparePrecision Vilnt32 Specify the de function uses Ivi SetAttrCon function, the I' comparing ca you set the at you specify, th The IVI engine differences be

> The value for higher the value callback to co

Valid Range: (

If you pass 0, which is 14.

You can modi[†] Ivi_SetAttrCon

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddRepeatedAttributeViSession

Usage

ViStatus Ivi_AddRepeatedAttributeViSession(ViSession vi, ViChar repeatedCapabilityName[], ViAttr newAttributeID, ViChar attributeName[], ViSession defaultValue, IviAttrFlags flags, ReadAttrViSession_CallbackPtr readCallback, WriteAttrViSession_CallbackPtr writeCallback);

Purpose

This function creates and configures a new ViSession attribute for the instrument session you specify. The value of the Repeated Capability parameter determines the repeated capability to which the attribute applies.

Parameters

Name

vi

Type ViSession

repeatedCapabilityName ViChar[]

newAttributeID

ViAttr

Description

The ViSessic identifies a p

Pass a string operate. For pass in the st

Pass the ID y distinct ID. Y(for the instrui begin with PF The include f

names for all includes attril class defines Each defined the specific ir

For each IVI ivi.h, except (example, ivi.) defines the fc

#define FL4

For each inst appears in th with the spec ividmm.h, def constant nan

#define FL4

For each spe constant nam an offset fron defines the fc

		#d FI 3L For nam IVI_ follo #d HI +	efine 45_ATTF) each attr ne in the (SPECIFI(wing con efine P34401A_ 1L)
attributeName	ViChar[]	The con:	name of stant narr
		For PRE	example, FIX_ATT
defaultValue	ViSession	Spe	cify the d
		The	IVI engir
			 IVI_A1 before IVI_VA The at before
flags	IviAttrFlags	Spe OR nev IVI_	cify the fl them tog er cachec _VAL_NE
		You and	can quei Ivi_SetA1
		Vali	d Values:
		Bit	Value
		0	0x0001
		1	0x0002
		2	0x0004

- 3 0x0008
- 4 0x0010
- 5 0x0020
- 6 0x0040
- 10 0x0400
- 11 0x0800
- 12 0x1000
- 13 0x2000
- 14 0x4000
- 15 0x8000

IVI_VAL_HII IVI_VAL_NO Use the IVI_` to access.

IVI_VAL_NO attribute but t

IVI_VAL_NO drivers can q value of the ϵ

IVI_VAL_NO drivers can rr the value of t

IVI_VAL_NO the value of t query the val

IVI_VAL_NO the value of t modify the va

IVI_VAL_NE[•] value of the *a* attribute. The attribute, if pr

IVI_VAL_AL' the attribute, attribute. IVI_VAL_MU value for eac

IVI_VAL_CO coerces value software. Do undocumente callback. Wh the IVI engine from the instr engine invoke function. Whe allow it to retain

- 1. The ir
- 2. If you instru two as the at value one o IVI_V VAL WA

IVI_VAL_WA the operation callback for t

IVI_VAL_WA the operation for the attribu

IVI_VAL_USI invoke the re mode.

IVI_VAL_DO the PREFIX_(driver and the enabled, the

calling the reengine never readCallback ReadAttrViSession_CallbackPtr Specify the re request the c You must def instrument dr ViStatus _VI_ repCapName, ViAttr attribut ViSession *va Upon entry to the callback, N Note 1 box to parame If you do not You can char Ivi_SetAttrRea writeCallback WriteAttrViSession_CallbackPtr Specify the w you set the a You must def instrument dr ViStatus _VI_ repCapName, $\overline{\mathbb{N}}$ Note 1 box to parame If you do not You can char Ivi_SetAttrWr

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddRepeatedAttributeViString

Usage

ViStatus Ivi_AddRepeatedAttributeViString(ViSession vi, ViChar repeatedCapabilityName[], ViAttr newAttributeID, ViChar attributeName[], ViChar defaultValue[], IviAttrFlags flags, ReadAttrViString_CallbackPtr readCallback, WriteAttrViString_CallbackPtr writeCallback);

Purpose

This function creates and configures a new ViString attribute for the instrument session you specify. The value of the Repeated Capability parameter determines the repeated capability to which the attribute applies.

Parameters

Name

vi

Type ViSession

repeatedCapabilityName ViChar[]

newAttributeID

ViAttr

Description

The ViSession identifies a part

Pass a string cooperate. For in: pass in the strir

Pass the ID you distinct ID. You for the instrume begin with PRE

The include file names for all of includes attribu class defines, *a* Each defined c the specific inst

For each IVI er ivi.h, except rej example, ivi.h (defines the folk

#define FL45_

For each instru appears in the with the specific ividmm.h, define constant name:

#define FL45_

For each specific constant name an offset from I defines the follo

		#d FL 3L	efine .45_ATT .)	R_J
		For nam IVI_ follo	each att ie in the SPECIFI wing coi	ribı dri [C_ nsta
		#d HI + (efine 234401A_ 1L)	_A_
attributeName	ViChar[]	The cons	name of stant nar	f th ne
		For PRE	example FIX_ATT	, if ΓR_
defaultValue	ViChar[]	Spe	cify the a	def
		The	IVI engi	ne
			• IVI_A before IVI_V	FTF 9 yc AL
			• The a before	ttril 9 yc
flags	IviAttrFlags	Spe OR neve IVI_	cify the f them tog er cache _VAL_NE	lac jetl d, 1 EVI
		You and	can que Ivi_SetA	ery Attri
		Valio	d Values	:
		Bit	Value	Fl
		0	0x0001	IV
		1	0x0002	IV
		2	0x0004	IV

- 3 0x0008 IV
- 4 0x0010 IV
- 5 0x0020 IV
- 6 0x0040 IV
- 10 0x0400 IV
- 11 0x0800 IV
- 12 0x1000 IV
- 13 0x2000 IV
- 14 0x4000 IV
- 15 0x8000 IV

IVI_VAL_HIDD IVI_VAL_NOT_ Use the IVI_VA to access.

IVI_VAL_NOT_ drivers can que value of the atti

IVI_VAL_NOT_ drivers can more the value of the

IVI_VAL_NOT_ the value of the query the value

IVI_VAL_NOT_ the value of the modify the valu

IVI_VAL_NEVI value of the atti attribute. The IV attribute, if pres

IVI_VAL_ALW¹ the attribute, if attribute.

IVI_VAL_MUL^{*}. value for each

IVI_VAL_COEF coerces values software. Do no undocumented callback. When the IVI engine i from the instrur engine invokes function. When allow it to retair

- 1. The inst
- 2. If you se instrume two ass the attril value th one or t IVI_VAI

IVI_VAL_WAIT the operation c callback for the

IVI_VAL_WAIT the operation c for the attribute

IVI_VAL_USE_ invoke the reac mode.

IVI_VAL_DON' the PREFIX_Ge driver and the I enabled, the IV

calling the read engine never tc ReadAttrViString_CallbackPtr readCallback Specify the rea request the cur You must define instrument drive ViStatus _VI_FL repCapName, Vi Unlike the read the current valu the current valu function. \mathbb{N} **Note** If y box to de paramete If you do not wa writeCallback WriteAttrViString_CallbackPtr Specify the writ you set the attr You must define instrument drive ViStatus _VI_FL repCapName, Vi \mathbb{N} **Note** If y box to de paramete If you do not wa You can change Ivi_SetAttrWrite

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddToChannelTable

Usage

ViStatus Ivi_AddToChannelTable(ViSession vi, ViChar ChannelStringsToBeAdded[]);

Purpose

This function adds additional channel strings to the channel table you establish with Ivi_BuildChannelTable.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
ChannelStringsToBeAdded	ViChar[]	Pass a string containing a the list of additional channel strings you want to add to the channel table. You must separate channel strings with commas. You can include spaces after the commas.
		For example, to add "3" and "4" as valid channel strings for the instrument session, pass "3, 4".

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_AddToRepCapTable

Usage

ViStatus Ivi_AddToRepCapTable(ViSession vi, ViChar repeatedCapabilityName[], ViChar Identifiers[]);

Purpose

This function adds additional repeated capability identifiers to the repeated capability table you establish with Ivi_BuildRepCapTable.
Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapabilityName	ViChar[]	Pass a string containing the name of the repeated capability on which to operate. For instance, if you are working with the table of channel names, pass in the string "Channel".
Identifiers	ViChar[]	Pass a string containing a list of additional repeated capability identifiers you want to add to the repeated capability table. You must separate repeated capability identifiers with commas. You can include spaces after the commas. For example, to add "3" and "4" as valid identifiers for the repeated capability, pass "3, 4".

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_Alloc

Usage

ViStatus Ivi_Alloc(ViSession vi, ViInt32 Memory_Block_Size, ViAddr* Memory_Block_Pointer);

Purpose

This function allocates memory for an object of the size you specify and initializes all bytes to zero. If you specify a non-NULL IVI session handle, the function associates the memory block with the session by inserting it into the list of memory blocks the IVI engine maintains for the session.

You can call Ivi_Free to free the memory block. You can call Ivi_FreeAll to free all of the memory blocks that you allocate for the session with Ivi_Alloc or Ivi_RangeTableNew. When you call Ivi_Dispose on the session, it calls Ivi_FreeAll for you.

If the function cannot allocate the space or you pass 0 for the Memory Block Size parameter, the function sets the Memory Block Pointer parameter to VI_NULL and returns an error.

Name	Туре	Description
vi	ViSession	If you want to associate the memory block with a particular IVI session, pass the IVI session handle that you obtain from Ivi_SpecificDriverNew. Otherwise, pass VI_NULL.
Memory_Block_Size	Vilnt32	Specify the number of bytes you want to allocate. You must pass a non-zero value.
Memory_Block_Pointer ViAd		Returns a pointer to the memory block the function allocates.
		If the function cannot allocate the space or you pass 0 for the Memory Block Size parameter, this parameter returns VI_NULL and the function returns an error.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_ApplyDefaultSetup

Usage

ViStatus Ivi_ApplyDefaultSetup(ViSession vi);

Purpose

This function applies the default attribute setup you specify in the lvi Configuration file. The specific driver should call this function after completely initializing a new session.

This function has no effect on the instrument session unless the application initializes the driver using an IVI class driver, or unless the application initializes the specific driver using an IVI logical name.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_AttachToConfigStoreHandle

Usage

ViStatus Ivi_AttachToConfigStoreHandle (IviConfigStoreHandle handle, ViBoolean discardExistingHandle);

Purpose

Normally, the IVI engine creates and owns an internal instance of the IVI Configuration Server. Use this function to specify that the IVI engine should use an IVI Configuration Server instance that you create and own. To use this function, you must first acquire a handle from the IVI Configuration Server C API.

During the period of time in which the IVI engine uses the handle that you specify in this function, changes to the IVI Configuration Store file on disk are not recognized by the IVI engine.

Use this function in the following situations:

- You want to dynamically create and/or destroy IVI configuration items in memory and have them used by the IVI engine, without first reading the contents of the IVI Configuration Store file.
- You want ensure that the IVI engine does not automatically refresh the configuration items in memory when the Master Configuration Store file is modified on disk.

Parameters		
Name	Туре	Description
handle	IviConfigStorehandle	The handle to the IVI Configuration Server instance that you want the IVI engine to use.
		Create this handle using the IVI Configuration Server C API.
discardExistingHandle	ViBoolean	Specifies whether to discard the IVI Configuration Server instance that the IVI engine uses internally.
		Pass VI_TRUE to discard the handle.
		Pass VI_FALSE to prevent the IVI engine from discarding the handle. This is useful if you are using an IVI Configuration Server instance that you obtained from a previous call to Ivi_GetConfigStoreHandle. If you pass VI_FALSE for this parameter, you must discard the handle yourself.
		Default Value: VI_TRUE

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_AttributeIsCached

Usage

ViStatus Ivi_AttributeIsCached(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViBoolean* cached)

Purpose;

This function indicates whether the IVI engine believes that the cache value of the attribute accurately reflects the state of the instrument.

The function returns VI_FALSE if the IVI_VAL_NEVER_CACHE flag for the attribute is set, there is no value in the cache, or the cache value has been invalidated.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID ViAttr	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the followinc constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

cached

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

ViBoolean* Indicates whether the IVI engine believes that the cache value of the attribute accurately reflects the state of the instrument.

Values:

(1) VI_TRUE:	Current cache value reflects the instrument state.
(0) VI_FALSE:	Current cache value might not reflect the instrument state.

This value is VI_FALSE if the IVI_VAL_NEVER_CACHE flag for the attribute is set, there is no value in the cache, or the cache value had been invalidated.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_BuildChannelTable

Usage

ViStatus Ivi_BuildChannelTable (ViSession vi, ViChar defaultChannelsStringsList[], ViBoolean allowUnknownChannelStrings, Ivi_ValidateChannelStringFunc chanStringValidationFunction);

Purpose

This function creates the initial channel table for an IVI session. A channel table consists of the channel strings that are valid for the instrument session. When you create attributes with the Ivi_AddAttribute functions, you set the IVI_VAL_MULTI_CHANNEL flag for attributes that have different values for each channel. You use this function to specify the set of channels.

You must call Ivi_BuildChannelTable in your PREFIX_IviInit function. If you call it again at a later point, it discards the old channel table and builds a new one. To add channel strings to an existing channel table, call Ivi_AddToChannelTable. To restrict an attribute to a subset of channels, call Ivi_RestrictAttrToChannels.

The IVI engine maintains the channel table for the session. If the user defines any virtual channel names in the configuration store, the IVI engine associates the virtual names with the entries in the table.

Name	Туре	Descriț
vi	ViSession	The ViS obtain fi Ivi_Spec handle i IVI sess
DefaultChannelsStringsList	ViChar[]	Pass a a list of represe instrume channel You car the com
		instrum and you "2" as th
allowUnknownChannelStrings	ViBoolean	Set this you war pass ch not in th your ins function VI_TRU you mus value fo Validatic In driver instrumo for this p
chanStringValidationFunction	Ivi_ValidateChannelStringFunc	The ivi.l the

Ivi_Valic typedef the chai valid ch callback VI_FAL: parame string re channel must re *isValid engine 1 string tc after the returns. initialize an IVI c engine a the defa channel channel table in: function this if yc channel attribute Ivi Add then cal Ivi Res every at exclude channel lvi_App function Ivi_App engine (pass VI Unknow parame

this para

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_BuildRepCapTable

Usage

ViStatus Ivi_BuildRepCapTable(ViSession vi, ViChar repeatedCapabilityName[], ViChar Identifiers[]);

Purpose

This function creates a repeated capability table for an IVI session. A repeated capability table consists of a name that identifies the repeated capability and the identifier strings that are valid for the instrument session. When you create attributes with the Ivi_AddRepeatedAttribute functions, you set the Repeated Capability parameter to the name of the repeated capability to which the attribute applies.

The repeated capability table must be created before attributes can be added for the repeated capability. To add identifier strings to an existing repeated capability table, call Ivi_AddToRepCapTable. To restrict an attribute to a subset of repeated capability identifiers, call Ivi_RestrictAttrToInstances. If you call this function more than once for the same repeated capability, the function will return the IVI_ERROR_REPEATED_CAPABILITY_ALREADY_EXISTS error.

The IVI engine maintains the repeated capability table for the session. If the user defines any virtual repeated capability names in the configuration store, the IVI engine associates the virtual names with the entries in the table.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapabilityName	ViChar[]	Pass a string containing the name of the repeated capability on which to operate. For instance, if you are working with the table of channel names, pass in the string "Channel".
Identifiers	ViChar[]	Pass a string containing a list of additional repeated capability identifiers you want to add to the repeated capability table. You must separate repeated capability identifiers with commas. You can include spaces after the commas. For example, to add "3" and "4" as valid identifiers for the repeated capability, pass "3, 4".

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_CheckAttributeViAddr

Usage

ViStatus Ivi_CheckAttributeViAddr(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViAddr attributeValue);

Purpose

This function checks the validity of a value you specify for a ViInt32 attribute. The function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error. If the attribute has no range table or check callback, the function assumes the value is valid.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.
IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViAddr Pass the value which you want to verify as a valid value for the attribute given the current settings of the instrument session. If the attribute currently showing in the Attribute ID ring control

has named constants as valid values, you can bring up a list of them on this control by pressing <ENTER>. Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckAttributeViBoolean

Usage

ViStatus Ivi_CheckAttributeViBoolean(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViBoolean attributeValue);

Purpose

This function checks the validity of a value you specify for a ViBoolean attribute. The function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If you provide a check callback for the attribute, the function invokes the check callback to validate the value. If the value is invalid, the function returns an error. If the attribute has no check callback, the function assumes the value is valid.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViBoolean Pass the value which yc want to verify as valid val for the attribute given the current settings the instrume session. If the attribute currently showing the Attribute ring con

has nam constant as valid values, y can brin up a list them on this cont by pressing <ENTEF Select a value by doubleclicking it or by highlight it and th pressing <ENTEF Some of the value might nc be valid dependi on the current settings the instrume session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckAttributeViInt32

Usage

ViStatus Ivi_CheckAttributeViInt32(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViInt32 attributeValue);

Purpose

This function checks the validity of a value you specify for a ViInt32 attribute. The function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error. If the attribute has no range table or check callback, the function assumes the value is valid.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

Vilnt32 Pass the value which you want to verify as a valid value for the attribute given the current settings of the instrument session. If the attribute currently showing in the Attribute ID ring control

has named constants as valid values, you can bring up a list of them on this control by pressing <ENTER>. Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckAttributeViInt64

Usage

ViStatus Ivi_CheckAttributeViInt64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViInt64 attributeValue);

Purpose

This function checks the validity of a value you specify for a ViInt64 attribute. The function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error. If the attribute has no range table or check callback, the function assumes the value is valid.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines

IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define
HP34401A_ATTR_TRIGGER_TYPE \

(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

optionFlags ViInt32 Use this parameter to request special behavior. In most cases, you pass 0. You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

Vilnt64 Pass the value which you want to verify as a valid value for the attribute given the current settings of the instrument session. If the attribute currently showing in the Attribute ID ring control

has named constants as valid values, you can bring up a list of them on this control by pressing <ENTER>. Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckAttributeViReal64

Usage

ViStatus Ivi_CheckAttributeViReal64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViReal64 attributeValue);

Purpose

This function checks the validity of a value you specify for a ViReal64 attribute. The function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error. If the attribute has no range table or check callback, the function assumes the value is valid.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.
IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViReal64 Pass the value which you want to verify as a valid valu for the attribute given the current settings c the instrumer session. If the attribute currently showing i the Attribute ring contr

has name constants as valid values, y can bring up a list c them on this contr by pressing <ENTER: Select a value by doubleclicking o it or by highlightii it and the pressing <ENTER: Some of the value might not be valid dependin on the current settings c the instrumer session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckAttributeViSession

Usage

ViStatus Ivi_CheckAttributeViSession(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViSession attributeValue);

Purpose

This function checks the validity of a value you specify for a ViSession attribute. The function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If you provide a check callback for the attribute, the function invokes the check callback to validate the value. If the value is invalid, the function returns an error. If the attribute has no check callback, the function assumes the value is valid.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViSession Pass the value which yc want to verify as valid val for the attribute given the current settings the instrume session. If the attribute currently showing the Attribute ring cont

has nam constant as valid values, y can bring up a list them on this cont by pressing <ENTEF Select a value by doubleclicking (it or by highlight it and the pressing <ENTEF Some of the value might nc be valid dependi on the current settings the instrume session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckAttributeViString

Usage

ViStatus Ivi_CheckAttributeViString(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViChar attributeValue[]);

Purpose

This function checks the validity of a value you specify for a ViString attribute. The function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If you provide a check callback for the attribute, the function invokes the check callback to validate the value. If the value is invalid, the function returns an error. If the attribute has no check callback, the function assumes the value is valid.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViChar[Pass the

1

value which you want to verify as a valid value for the attribute given the current settings of the instrument session. If the attribute currently showing in the Attribute ID ring contro

has named constants as valid values, you can bring up a list of them on this control by pressing <ENTER>. Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckBooleanRange

Usage

ViStatus Ivi_CheckBooleanRange(ViBoolean value, ViStatus errorCode);

Purpose

This function verifies that the ViBoolean value you specify is either VI_TRUE (1) or VI_FALSE (0).

If the value is not VI_TRUE or VI_FALSE, the function returns the error code you specify.

Parameters

Name
valueTypeDescriptionViBooleanSpecify the value you want to check.errorCodeViStatusSpecify the error code the function returns if the
value you specify is not VI_TRUE or VI_FALSE.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_CheckNumericRange

Usage

ViStatus Ivi_CheckNumericRange(ViReal64 value, ViReal64 minimum, ViReal64 maximum, ViStatus errorCode);

Purpose

This function verifies that a ViInt32 or ViReal64 value falls within the a range you specify.

If it the value does not fall with the range, the function returns the error code you specify. The range is inclusive. In other words, the function returns the error code if the value is less than the minimum value or greater than the maximum value. When you use this function on a parameter to a user-callable function in your instrument driver, use the viCheckParm macro around this function.

The following example shows how to use the viCheckParm macro around this function:

viCheckParm(Ivi_CheckNumericRange(parmVal,min,max,errorCode) ,
parmPosition, parmName);

In this example, **parmPosition** is the 1-based position of the parameter in the parameter list of the user-callable function, and **parmName** is a string that contains the name of the parameter. Ivi_CheckNumericRange stores the **errorCode** you pass to it as the primary error code. viCheckParm converts the parmPosition to one of the VXI*plug&play* error codes for invalid parameters and stores it as the secondary error code.

It stores the **parmName** as the error elaboration string.

Parameters

Name	Туре	Description
value	ViReal64	Specify the value you want to check.
minimum	ViReal64	Specify the minimum value of the range.
maximum	ViReal64	Specify the maximum value of the range.
errorCode	ViStatus	Specify the error code the function returns if the value you specify does not fall within the range you specify.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_ClearErrorInfo

Usage

ViStatus Ivi_ClearErrorInfo(ViSession vi);

Purpose

This function clears the error information for the current execution thread and the IVI session you specify. If you pass VI_NULL for the vi parameter, this function clears the error information only for the current execution thread.

Instrument drivers export this function to the end-user through the PREFIX_ClearError function. Normally, the error information describes the first error that occurred since the end-user last called PREFIX_GetErrorInfo or PREFIX_ClearError.

The error information includes a primary error code, secondary code error, and an error elaboration string. For a particular session, this information is the same as the values held in the following attributes:

IVI_ATTR_PRIMARY_ERROR or PREFIX_ATTR_PRIMARY_ERROR IVI_ATTR_SECONDARY_ERROR or PREFIX_ATTR_SECONDARY_ERROR IVI_ATTR_ERROR_ELABORATION or PREFIX_ATTR_ERROR_ELABORATION

The IVI engine also maintains this error information separately for each thread. This is useful if you do not have a session handle to pass to Ivi_SetErrorInfo or Ivi_GetErrorInfo, which occurs when a call to Ivi_SpecificDriverNew fails.

This function sets the primary and secondary error codes to VI_SUCCESS (0), and sets the error elaboration string to ".

Avoid calling this function except to implement the PREFIX_ClearError function. Normally, it is the responsibility of the end-user to decide when to clear the error information. Ivi_GetErrorInfo, which the end-user calls through PREFIX_GetErrorInfo, always clears the error information.

Parameters

Name Type Description

vi ViSession To clear the error information for a particular IVI session, pass the ViSession handle that you obtain from Ivi_SpecificDriverNew. When you pass a ViSession handle, the function also clears the error information for the current thread.

To clear only the error information for the current thread, pass VI_NULL.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_ClearInstrSpecificErrorQueue

Usage

ViStatus Ivi_ClearInstrSpecificErrorQueue(ViSession vi);

Purpose

This function removes all entries from the instrument-specific error queue.
Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_ClearInterchangeWarnings

Usage

ViStatus Ivi_ClearInterchangeWarnings(ViSession vi);

The specific driver performs interchangeability checking if the IVI_ATTR_INTERCHANGE_CHECK attribute is set to VI_TRUE. This function clears the list of current interchange warnings.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_CoerceBoolean

Usage

ViStatus Ivi_CoerceBoolean(ViBoolean* value);

This function coerces a value you specify to a valid ViBoolean value. If the value is non-zero, the function changes it to $VI_TRUE(1)$.

Parameters					
Name	Туре	Description			
value	ViBoolean*	Pass the address of the value you want to coerce. If the value is non-zero, the function changes it to VI_TRUE (1).			

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_CoerceChannelName

Usage

ViStatus Ivi_CoerceChannelName(ViSession vi, ViChar ChannelName[], ViConstString* ChannelString);

If your driver supports multiple channels, you must call Ivi_CoerceChannelName in driver functions that use the channel string directly. When you call an Ivi_SetAttribute, Ivi_GetAttribute, or Ivi_CheckAttribute function. the IVI engine calls Ivi_CoerceChannelName internally before invoking the read, write, check, coerce, compare, and range table callback functions.

To be valid, Channel Name must be one of the following:

- VI_NULL, in which case the function sets the Channel String parameter to VI_NULL.
- An empty string, in which case the function returns the address of that empty string in the Channel String parameter.
- A specific driver channel string. The specific instrument driver specifies the valid channel strings using Ivi_BuildChannelTable or Ivi_AddToChannelTable. If you pass one of these strings, the function returns the address of the channel string in the channel table that the IVI engine maintains for the session.
- A virtual channel name that the end-user specifies in the configuration store. Virtual channel names are valid only if the end-user opens the session from a class driver and assigns a valid specific driver channel string to the virtual name in the configuration store. If you pass a valid virtual name to this function, the function returns the address of the corresponding specific driver channel string in the channel table that the IVI engine maintains for the session.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
ChannelName	ViChar[]	Pass the channel name that you want to verify and convert to a specific driver channel name string. Usually, this is the string that the end-user passes into a specific driver function.
		The parameter accepts VI_NULL, an empty string, a specific driver-defined channel name identifier, or a virtual channel name identifier.
ChannelString	ViConstString*	Returns a pointer to a channel string in the channel table that the IVI engine maintains for the session.
		Do not modify the contents of the channel string.
		If you pass VI_NULL in the Channel Name Identifier parameter, the function returns VI_NULL in this parameter. If you pass an empty string, the function returns the address of the empty string.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_CoerceRepCapName

Usage

ViStatus Ivi_CoerceRepCapName(ViSession vi, ViChar repeatedCapabilityName[], ViChar repeatedCapabilityIdentifier[], ViConstString* repeatedCapabilityString);

This function verifies that the Repeated Capability Name parameter you pass is valid and returns a pointer to the corresponding specific driver repeated capability string in the Repeated Capability String output parameter.

If your driver supports multiple repeated capability instances, you must call Ivi_CoerceRepCapName in driver functions that use the string directly. When you call an Ivi_SetAttribute, Ivi_GetAttribute, or Ivi_CheckAttribute function, the IVI engine coerces the name internally before invoking the read, write, check, coerce, compare, and range table callback functions.

To be valid, Repeated Capability Name must be one of the following:

- VI_NULL, in which case the function sets the Repeated Capability String parameter to VI_NULL.
- An empty string, in which case the function returns the address of that empty string in the Repeated Capability String parameter.
- A specific driver repeated capability string. The specific instrument driver specifies the valid repeated capability strings using Ivi_BuildRepCapTable or Ivi_AddToRepCapTable. If you pass one of these strings, the function returns the address of the repeated capability string in the repeated capability table that the IVI engine maintains for the session.
- A virtual repeated capability name that the end-user specifies in the configuration file. Virtual repeated capability names are valid only if the the end-user opens the session from a class driver and assigns a valid specific driver repeated capability string to the virtual name in the configuration file. If you pass a valid virtual name to this function, the function returns the address of the corresponding specific driver repeated capability string in the repeated capability table that the IVI engine maintains for the session.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapabilityName	ViChar[]	Pass a string containing the name of the repeated capability on which to operate. For instance, if you are working with the table of channel names, pass in the string "Channel".
repeatedCapabilityIdentifier	ViChar[]	Pass the repeated capability name that you want to verify and convert to a specific driver repeated capability string. Usually, this is the string that the end-user passes into a specific driver function.
		The parameter accepts VI_NULL, an empty string, a specific driver- defined repeated capability identifier, or a virtual repeated capability identifier.
repeatedCapabilityString	ViConstString*	Returns a pointer to a repeated capability

identifier string in the repeated capability table that the IVI engine maintains for the session.

Do not modify the contents of the repeated capability string.

If you pass VI_NULL in the Repeated Capability Identifier parameter, the function returns VI_NULL in this parameter. If you pass an empty string, the function returns the address of the empty string.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_CompareWithPrecision

Usage

ViStatus Ivi_CompareWithPrecision(ViInt32 digitsofPrecision, ViReal64 a, ViReal64 b, ViInt32* result);

This function compares two ViReal64 values using the number of decimal digits of precision you specify.

If the two values are not exactly equal, the function uses the following logic, where a and b are the values you want to compare, and d is the digits of precision you specify.

```
if a == 0
-(d-1)
if |b| < 10 then a == b.
else /* a != 0 */
|a-b| -(d-1)
if ----- < 10 then a == b
|a|
```

The function returns the following values.

```
0 if a == b
-1 if a < b
1 if a > b
```

Name	Туре	Description
digitsofPrecision	Vilnt32	Specify the number of decimal digits of precision you want to use to compare the two ViReal64 values.
		Valid Range: 0, or 1 to 14
		If you pass 0, the function sets the precision to the IVI default for this value, which is 14.
a	ViReal64	Specify the first value you want to compare.
b	ViReal64	Specify the second value you want to compare.
result	Vilnt32*	Returns the results of the comparison. Return values: 0 (if $a == b$) 1 (if $a > b$) -1 (if $a < b$)

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_DefaultCheckCallbackViInt32

Usage

ViStatus Ivi_DefaultCheckCallbackViInt32(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 attributeValue);

This function performs the default actions for checking the validity of a ViInt32 attribute value. The IVI engine automatically installs this callback when you call Ivi_AddAttributeViInt32.

If you want to add to the actions of this callback, install your own callback with Ivi_SetAttrCheckCallbackViInt32, and call this function from your callback.

This function does the following:

- 1. Calls Ivi_GetAttrRangeTable to obtain the range table for the attribute. If the range table is invalid, the function returns an error. If there is no range table, the function returns VI_SUCCESS.
- 2. Calls Ivi_GetViInt32EntryFromValue to find an entry that matches the value.
- 3. Returns VI_SUCCESS if it can find an entry. Otherwise it returns an error.

Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

attributeValue

Vilnt32

Specify the value you want to validate.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_DefaultCheckCallbackViInt64

Usage

ViStatus Ivi_DefaultCheckCallbackViInt64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt64 attributeValue);

This function performs the default actions for checking the validity of a ViInt64 attribute value. The IVI engine automatically installs this callback when you call Ivi_AddAttributeViInt64.

If you want to add to the actions of this callback, install your own callback with Ivi_SetAttrCheckCallbackViInt64, and call this function from your callback.

This function does the following:

- 1. Calls Ivi_GetAttrRangeTable to obtain the range table for the attribute. If the range table is invalid, the function returns an error. If there is no range table, the function returns VI_SUCCESS.
- 2. Calls Ivi_GetViInt64EntryFromValue to find an entry that matches the value.
- 3. Returns VI_SUCCESS if it can find an entry. Otherwise it returns an error.

Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines

IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define
HP34401A_ATTR_TRIGGER_TYPE \

(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

attributeValue

Vilnt64

Specify the value you want to validate.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_DefaultCheckCallbackViReal64

Usage

ViStatus Ivi_DefaultCheckCallbackViReal64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViReal64 attributeValue);

This function performs the default actions for checking the validity of a ViReal64 attribute value. The IVI engine automatically installs this callback when you call Ivi_AddAttributeViReal64.

If you want to add to the actions of this callback, install your own callback with Ivi_SetAttrCheckCallbackViReal64, and call this function from your callback.

This function does the following:

- 1. Calls Ivi_GetAttrRangeTable to obtain the range table for the attribute. If the range table is invalid, the function returns an error. If there is no range table, the function returns VI_SUCCESS.
- 2. Calls Ivi_GetViReal64EntryFromValue to find an entry that matches the value.
- 3. Returns VI_SUCCESS if it can find an entry. Otherwise it returns an error.

Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.
Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

attributeValue

ViReal64

⁴ Specify the value you want to validate.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_DefaultCoerceCallbackViBoolean

Usage

ViStatus Ivi_DefaultCoerceCallbackViBoolean(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViBoolean attributeValue, ViBoolean* coercedValue);

Purpose

This function performs the default actions for coercing a value for a ViBoolean attribute. The IVI engine automatically installs this callback when you call Ivi_AddAttributeViBoolean.

You can install your own coerce callback by calling Ivi_SetAttrCoerceCallbackViBoolean.

This function sets the Coerced Value parameter to VI_TRUE (1) if the value you specify as the Attribute Value parameter is non-zero.



Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.

Name	Туре	Description
vi ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.	
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the followinc constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define
HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE
+ 1L)

attributeValue ViBoolean Specify the value you want to coerce.

coercedValue ViBoolean* Returns the coerced value.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_DefaultCoerceCallbackViInt32

Usage

ViStatus Ivi_DefaultCoerceCallbackViInt32(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 attributeValue, ViInt32* coercedValue);

Purpose

This function performs the default actions for coercing a value for a ViInt32 attribute. The IVI engine automatically installs this callback when you call Ivi_AddAttributeViInt32.

You can install your own coerce callback by calling Ivi_SetAttrCoerceCallbackViInt32.

This function does the following:

- 1. Calls Ivi_GetAttrRangeTable to obtain the range table for the attribute. If the range table is invalid, the function returns an error.
- 2. If there is no range table or its type is not IVI_VAL_COERCED, the function sets the Coerced Value parameter to the value you passed in as the Attribute Value parameter.
- 3. Calls Ivi_GetViInt32EntryFromValue to find an entry that matches the value.
- 4. If it can find an entry, it sets Coerced Value to the coercedValue field in the range table entry and returns VI_SUCCESS. Otherwise it returns an error.
- Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE.

For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define
HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE
+ 1L)

attributeValue Vilnt32 Specify the value you want to coerce.

coercedValue Vilnt32* Returns the value to which the function

coerces the input value based on the range table for the attribute.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_DefaultCoerceCallbackViInt64

Usage

ViStatus Ivi_DefaultCoerceCallbackViInt64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt64 attributeValue, ViInt64* coercedValue);

Purpose

This function performs the default actions for coercing a value for a ViInt64 attribute. The IVI engine automatically installs this callback when you call Ivi_AddAttributeViInt64.

You can install your own coerce callback by calling Ivi_SetAttrCoerceCallbackViInt64.

This function does the following:

- 1. Calls Ivi_GetAttrRangeTable to obtain the range table for the attribute. If the range table is invalid, the function returns an error.
- 2. If there is no range table or its type is not IVI_VAL_COERCED, the function sets the Coerced Value parameter to the value you passed in as the Attribute Value parameter.
- 3. Calls Ivi_GetViInt64EntryFromValue to find an entry that matches the value.
- 4. If it can find an entry, it sets Coerced Value to the coercedValue field in the range table entry and returns VI_SUCCESS. Otherwise it returns an error.
- Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one o the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines

IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define
HP34401A_ATTR_TRIGGER_TYPE \

(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

attributeValue	Vilnt64	Specify the value you want to coerce.
coercedValue	Vilnt64*	If the search succeeds, this parameter returns the value of the coercedValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_DefaultCoerceCallbackViReal64

Usage

ViStatus Ivi_DefaultCoerceCallbackViReal64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViReal64 attributeValue, ViReal64* coercedValue);

Purpose

This function performs the default actions for coercing a value for a ViReal64 attribute. The IVI engine automatically installs this callback when you call Ivi_AddAttributeViReal64.

You can install your own coerce callback by calling Ivi_SetAttrCoerceCallbackViReal64.

This function does the following:

- 1. Calls Ivi_GetAttrRangeTable to obtain the range table for the attribute. If the range table is invalid, the function returns an error.
- 2. If there is no range table or its type is not IVI_VAL_COERCED, the function sets the Coerced Value parameter to the value you passed in as the Attribute Value parameter.
- 3. Calls Ivi_GetViReal64EntryFromValue to find an entry that matches the value.
- 4. If it can find an entry, it sets Coerced Value to the coercedValue field in the range table entry and returns VI_SUCCESS. Otherwise it returns an error.
- Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define
HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE
+ 1L)

attributeValue ViReal64 Specify the value you want to coerce.

coercedValue ViReal64* Returns the value to which the function

coerces the input value based on the range table for the attribute.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_DefaultCompareCallbackViReal64

Usage

ViStatus Ivi_DefaultCompareCallbackViReal64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViReal64 a, ViReal64 b, ViInt32* result);

Purpose

This function performs the default compare actions for a ViReal64 attribute. The IVI engine invokes the compare callback to compare the cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

The IVI engine automatically installs this callback when you call Ivi_AddAttributeViReal64. The IVI engine installs the default compare callback rather than comparing based on strict equality because of differences between computer and instrument floating point representations.

You can install your own compare callback by calling Ivi_SetAttrCompareCallbackViReal64.

If the two values are not exactly equal, the function uses the following logic, where a and b are the values you want to compare, and d is the digits of precision you specify when you call Ivi_AddAttributeViReal64 or Ivi_SetAttrComparePrecision.

if a == 0 -(d-1) if |b| < 10 then a == b. else /* a !=



Note Do not call this function directly unless you are calling it from your own callback or you have already called Ivi_LockSession.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

ViReal64 Specify the first value to compare. Normally, this is the new value to which you are trying to set the attribute. If

a

		there is a coerce callback or coerced range table for the attribute, the IVI engine has already coerced the value.
b	ViReal64	Specify the second value to compare. Normally, this is the current cache value of the attribute.
result	Vilnt32*	Returns the results of the comparison. Return values: 0 (if $a == b$) 1 (if $a > b$) -1 (if $a < b$)

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_DeleteAttribute

Usage

ViStatus Ivi_DeleteAttribute(ViSession vi, ViAttr attributeID);

Purpose

This function deletes the attribute you specify. Typically, it is not necessary for you to call this function.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user- accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:
		#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define FL45_ATTR_HOLD_THRESHOLD \
(IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)
Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_DeleteAttributeInvalidation

Usage

ViStatus Ivi_DeleteAttributeInvalidation(ViSession vi, ViAttr attributeID, ViAttr dependentAttributeID);

This function removes the invalidation dependency relationship between two attributes. You establish invalidation dependency relationships using Ivi_AddAttributeInvalidation.

Para	ame	ters
------	-----	------

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrume driver defines constant names for all o the user-accessible attributes that app to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrume prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the followin constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, th specific driver include file uses the same constant name that appears in t instrument class include file, except th the specific instrument prefix replaces

the class prefix. For example, the DMI class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value th is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the followi constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BAS + 3L)

For each attribute that is private to an instrument driver, the instrument drive source file defines a constant name ar assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BA) + 1L)

dependentAttributeID ViAttr Pass the ID of the attribute which is invalidated when the value of the othe attribute changes.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_DequeueInstrSpecificError

Usage

ViStatus Ivi_DequeueInstrSpecificError(ViSession vi, ViInt32* instrumentError, ViChar errorMessage[]);

This function retrieves the error code and description string from the oldest entry in the instrument-specific error queue. It also removes the entry from the queue.

Use the instrument-specific error queue if querying the instrument for its status causes the instrument to lose the error value. In your check status callback, call Ivi_QueueInstrSpecificError to insert the instrument error code in the queue, and then return the IVI_ERROR_INSTR_SPECIFIC error code from the callback. In your PREFIX_error_query function, call Ivi_InstrSpecificErrorQueueSize to determine if there is an error in the queue. If not, invoke the check status callback directly. In either case, if there is an error, call Ivi_DequeueInstrSpecificError to retrieve it.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
instrumentError	or Vilnt32*	Returns the error code from the oldest entry in the instrument-specific error queue.
		If you are not interested in this value, pass VI_NULL.
errorMessage	ViChar[]	Returns the error message from the oldest entry in the instrument-specific error queue.
		If you are not interested in this value, pass VI_NULL. Otherwise, pass a ViChar array that contains at least IVI_MAX_MESSAGE_BUF_SIZE (256) bytes.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

lvi_Dispose

Usage

ViStatus Ivi_Dispose(ViSession vi);

This function destroys the IVI session, all of its attributes, and the memory resources it uses.

This function does NOT close the instrument I/O session. You must do that yourself before calling this function.

You must unlock the session before calling Ivi_Dispose.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_DisposeInvalidationList

Usage

void = Ivi_DisposeInvalidationList(IviInvalEntry* invalidationList);

This function deallocates an invalidation list you obtain from Ivi_GetInvalidationList.

NameTypeDescriptioninvalidationListIviInvalEntry*Specify the pointer to an invalidation list
you obtain from Ivi_GetInvalidationList.

Ivi_DisposeLogicalNamesList

Usage

void = Ivi_DisposeLogicalNamesList(IviLogicalNameEntry*
logicalNamesList);

This function deallocates an invalidation list you obtain from Ivi_GetLogicalNamesList.

Name	Туре	Description
logicalNamesList	IviLogicalNameEntry*	Specify the pointer to the logical names list you obtain from Ivi_GetLogicalNamesList.

Ivi_Free

Usage

ViStatus Ivi_Free(ViSession vi, ViAddr Memory_Block_Pointer);

This function deallocates a memory block you allocate with Ivi_Alloc. If you specify a non-NULL IVI session handle, the function also removes the memory block from the list of memory blocks that the IVI engine maintains for the session.

For the vi parameter, you must specify the same IVI session handle that you pass to Ivi_Alloc when you allocate the memory block.

Para	met	ers
------	-----	-----

Name	Туре	Description
vi ViSession	Specify the same IVI session handle that you pass to Ivi_Alloc when you allocate the memory block.	
		If you pass VI_NULL for the vi parameter to Ivi_Alloc, pass VI_NULL for the vi parameter to this function.
memoryBlockPointer	ViAddr	This function deallocates a memory block you allocate with Ivi_Alloc. If you specify a non-NULL IVI session handle, the function also removes the memory block from the list of memory blocks that the IVI engine maintains for the session.
	For the vi parameter, you must specify the same IVI session handle that you pass to Ivi_Alloc when you allocate the memory block.	

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

lvi_FreeAll

Usage

ViStatus Ivi_FreeAll(ViSession vi);

This function deallocates all memory blocks you allocate with Ivi_Alloc or Ivi_RangeTableNew for the session.

When you call Ivi_Dispose on a session, it calls Ivi_FreeAll for you.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetAttrComparePrecision

Usage

ViStatus Ivi_GetAttrComparePrecision(ViSession vi, ViAttr attributeID, ViInt32* ComparePrecision);

This function returns the degree of decimal precision the default IVI compare callback currently uses for this attribute. For more information on the comparison precision, refer to Ivi_SetAttrComparePrecision.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID Vi	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM

class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

ComparePrecision ViInt32* The degree of precision the default IVI compare callback currently uses for this attribute.

The value is in terms of decimal digits of precision.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetAttributeFlags

Usage

ViStatus Ivi_GetAttributeFlags(ViSession vi, ViAttr attributeID, IviAttrFlags* flags);

This function obtains the current values of the flags for the attribute you specify.

Name	Туре	Description		
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a passion.		
attributeID	ViAttr	Pass the ID of an attribute for this parameter.		
		The include file for a specific instrument driver d constant names for all of the user-accessible att apply to the driver. This includes attributes that t defines, attributes that the instrument class defin attributes that are specific to the particular instru defined constant name begins with PREFIX_AT_PREFIX is the specific instrument prefix.		
		For each IVI engine attribute, the specific driver uses the same constant name that appears in is the specific instrument prefix replaces the IVI pr example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines the following constant		
		#define FL45_ATTR_CACHE IVI_ATTR_CACH		
		For each instrument class attribute, the specific file uses the same constant name that appears instrument class include file, except that the specific file instrument prefix replaces the class prefix. For ϵ DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the constant name:		
		#define FL45_ATTR_RANGE IVIDMM_ATTR_		
		For each specific instrument attribute, the specific include file defines a constant name and assign is an offset from IVI_SPECIFIC_PUBLIC_ATTR_example, fl45.h defines the following constant national statement of the following constant of the following constant statement of the following constant st		
		#define FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)		
flama		For instr assi IVI_ hp3 ² #d (IV	each atti rument d gns a va SPECIFI 401a.c d efine HP3 /I_SPEC	ribute that is private to an instrumer driver source file defines a constant i alue that is an offset from IC_PRIVATE_ATTR_BASE. For exan defines the following constant name: 34401A_ATTR_TRIGGER_TYPE \ CIFIC_PRIVATE_ATTR_BASE + 1L)
-------	---------------	---	--	---
nags	IVIAttr-lags^	Reti	urns the	current values of the flags of the att
		Vali	d Values	nays as bits.
		Bit	Value	Flag
		0	0x0001	IVI_VAL_NOT_SUPPORTED
		1	0x0002	IVI_VAL_NOT_READABLE
		2	0x0004	IVI_VAL_NOT_WRITABLE
		3	0x0008	IVI_VAL_NOT_USER_READABLE
		4	0x0010	IVI_VAL_NOT_USER_WRITABLE
		5	0x0020	IVI_VAL_NEVER_CACHE
		6	0x0040	IVI_VAL_ALWAYS_CACHE
		10	0x0400	IVI_VAL_MULTI_CHANNEL
		11	0x0800	IVI_VAL_COERCEABLE_ONLY_B
		12	0x1000	IVI_VAL_WAIT_FOR_OPC_BEFO
		13	0x2000	IVI_VAL_WAIT_FOR_OPC_AFTEF
		14	0x4000	IVI_VAL_USE_CALLBACKS_FOR
		15	0x8000	IVI_VAL_DONT_CHECK_STATUS
		See Ivi_/	the cont AddAttrit	trol help for the Flags parameter to t bute functions for detailed informatio

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeName

Usage

ViStatus Ivi_GetAttributeName(ViSession vi, ViAttr attributeID, ViChar nameBuffer[], ViInt32 bufferSize);

Purpose

This function obtains the name of an attribute.

Parameters

Name	Туре	Description ¹ The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.	
vi	ViSession		
attributeID	ViAttr	Pass the ID of an attribute for this parameter.	
		The include file for a specific instrument driver defines constant names for all of the user- accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.	
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:	
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE	
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:	
		#define FL45_ATTR_RANGE	

IVIDMM_ATTR_RANGE

		For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:
		#define FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)
		For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:
		#define HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)
nameBuffer ∨	'iChar[]	A buffer into which the function copies the name of the attribute.
bufferSize ∨	'ilnt32	Specify the number of bytes in the ViChar array you pass for the Name Buffer parameter.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeType

Usage

ViStatus Ivi_GetAttributeType(ViSession vi, ViAttr attributeID, IviValueType* dataType);

Purpose

This function obtains the data type of an attribute.

Parameters

Name	Туре	Description	
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.	
attributeID	ViAttr	Pass the ID of an attribute for this parameter.	
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that apply to the local defines, attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.	
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:	
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE	
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant	

name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define FL45_ATTR_HOLD_THRESHOLD \
(IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE +
1L)

dataType IviValueType* Returns the data type of the attribute you specify.

Values:

- (1) IVI_VAL_INT32 ViInt32
- (4) IVI_VAL_REAL64 ViReal64
- (5) IVI_VAL_STRING ViString
- (10) IVI_VAL_ADDR ViAddr
- (11) IVI_VAL_SESSION ViSession
- (13) IVI_VAL_BOOLEAN ViBoolean

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeViAddr

Usage

ViStatus Ivi_GetAttributeViAddr(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViAddr* attributeValue);

Purpose

This function obtains the current value of the ViInt32 attribute you specify. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is readable. If not, the function returns an error.
- 2. If the attribute cache value is currently valid, the read callback for the attribute is VI_NULL, or the IVI_ATTR_SIMULATE attribute is enabled and the IVI_ATTR_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function returns the cache value.
- 3. If the IVI_VAL_WAIT_FOR_OPC_BEFORE_READS flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 4. The function invokes the read callback for the attribute. Typically, the callback performs instrument I/O to obtain a new value. The IVI engine stores the new value in the cache.
- 5. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Parameters		
Name	Туре	Description
vi	ViSession	Returns a ViSession handle that you use to identify the session in subsequent function calls.
		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.
		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 same session in multiple execution threads. You can use functions Ivi_LockSession and Ivi_UnlockSession to protect sections of code that require exclusive access to the resource.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle

this ring control to the manual input box so you can type the ID constant.

If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <ENTER>, <SPACEBAR>, or <CTRL-DOWN>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <ENTER>, the dialog disappears and your choice appears in this function panel control.

If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you

can bring up a list of them by moving to the Attribute Value control and pressing <ENTER>.

optionFlagsViInt32Use this parameter to request special
behavior. In most cases, you pass 0.You can specify individual bits to

request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViAddr* Returns the current value of the attribute. Pass the address of a ViAddr variable. If the attribute currently showing in the Attribute IC ring contro has namec constants as valid values, you can bring

up a list of them on this control by pressing <ENTER>. Select a value by doubleclicking on it or by highlightinç it and then pressing <ENTER>.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeViBoolean

Usage

ViStatus Ivi_GetAttributeViBoolean(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViBoolean* attributeValue);

Purpose

This function obtains the current value of the ViBoolean attribute you specify. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is readable. If not, the function returns an error.
- 2. If the attribute cache value is currently valid, the read callback for the attribute is VI_NULL, or the IVI_ATTR_SIMULATE attribute is enabled and the IVI_ATTR_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function returns the cache value.
- 3. If the IVI_VAL_WAIT_FOR_OPC_BEFORE_READS flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 4. The function invokes the read callback for the attribute. Typically, the callback performs instrument I/O to obtain a new value. The IVI engine stores the value new in the cache.
- 5. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViBoolean* Returns the curi value o the attribute Pass th addres а ViBoole variable If the attribute current showin the Attribut ring cor has nar constar as valic values,

can brir up a lis them or this cor by pressin <ENTE Select (value b doubleclicking it or by highligh it and th pressin . <ENTE

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeViInt32

Usage

ViStatus Ivi_GetAttributeViInt32(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViInt32* attributeValue);

Purpose

This function obtains the current value of the ViReal64 attribute you specify. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is readable. If not, the function returns an error.
- 2. If the attribute cache value is currently valid, the read callback for the attribute is VI_NULL, or the IVI_ATTR_SIMULATE attribute is enabled and the IVI_ATTR_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function returns the cache value.
- 3. If the IVI_VAL_WAIT_FOR_OPC_BEFORE_READS flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 4. The function invokes the read callback for the attribute. Typically, the callback performs instrument I/O to obtain a new value. The IVI engine stores the new value in the cache.
- 5. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.
IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

Vilnt32* Returns the current value of the attribute. Pass the address of a Vilnt32 variable. If the attribute currently showing in the Attribute IE ring contro has named constants as valid values, you can bring

up a list of them on this contro by pressing <ENTER> Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeViInt64

Usage

ViStatus Ivi_GetAttributeViInt64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViInt64* attributeValue);

Purpose

This function obtains the current value of the ViReal64 attribute you specify. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is readable. If not, the function returns an error.
- 2. If the attribute cache value is currently valid, the read callback for the attribute is VI_NULL, or the IVI_ATTR_SIMULATE attribute is enabled and the IVI_ATTR_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function returns the cache value.
- 3. If the IVI_VAL_WAIT_FOR_OPC_BEFORE_READS flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 4. The function invokes the read callback for the attribute. Typically, the callback performs instrument I/O to obtain a new value. The IVI engine stores the new value in the cache.
- 5. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines

IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define
HP34401A_ATTR_TRIGGER_TYPE \

(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

optionFlags ViInt32 Use this parameter to request special behavior. In most cases, you pass 0. You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

Vilnt64* Returns the current value of the attribute. Pass the address of a Vilnt64 variable. If the attribute currently showing in the Attribute IE ring contro has named constants as valid values, you can bring

up a list of them on this contro by pressing <ENTER> Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeViReal64

Usage

ViStatus Ivi_GetAttributeViReal64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViReal64* attributeValue);

Purpose

This function obtains the current value of the ViReal64 attribute you specify. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is readable. If not, the function returns an error.
- 2. If the attribute cache value is currently valid, the read callback for the attribute is VI_NULL, or the IVI_ATTR_SIMULATE attribute is enabled and the IVI_ATTR_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function returns the cache value.
- 3. If the IVI_VAL_WAIT_FOR_OPC_BEFORE_READS flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 4. The function invokes the read callback for the attribute. Typically, the callback performs instrument I/O to obtain a new value. The IVI engine stores the new value in the cache.
- 5. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViReal64* Returns

the curre value of the attribute Pass the address a ViReal variable. If the attribute currently showing the Attribute ring cont has nam constant as valid values, y can brine

up a list them on this cont by pressing <ENTEF Select a value by doubleclicking (it or by highlight it and the pressing <ENTEF

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeViSession

Usage

ViStatus Ivi_GetAttributeViSession(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViSession* attributeValue);

Purpose

This function obtains the current value of the ViSession attribute you specify. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is readable. If not, the function returns an error.
- 2. If the attribute cache value is currently valid, the read callback for the attribute is VI_NULL, or the IVI_ATTR_SIMULATE attribute is enabled and the IVI_ATTR_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function returns the cache value.
- 3. If the IVI_VAL_WAIT_FOR_OPC_BEFORE_READS flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 4. The function invokes the read callback for the attribute. Typically, the callback performs instrument I/O to obtain a new value. The IVI engine stores the new value in the cache.
- 5. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViSession* Returns the curr value o the attribute Pass th address а ViSessi variable If the attribute currentl showing the Attribut ring cor has nar constar as valid values,

can brir up a list them or this con by pressin <ENTE Select ¿ value b doubleclicking it or by highligh it and th pressin . <ENTE

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttributeViString

Usage

ViStatus Ivi_GetAttributeViString(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViInt32 bufferSize, ViChar attributeValue[]);

Purpose

This function obtains the current value of the ViString attribute you specify.

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 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. 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.

Remember that the checkErr and viCheckErr macros ignore positive return values. If you use one of these macros around a call to this function, you lose the required buffer size when the function returns it. To retain this information, declare a separate local variable to store the required buffer size, and use the macro around the assignment of the return value to the local variable. The following is an example:

ViStatus error = VI_SUCCESS;ViInt32 requiredBufferSize;

checkErr(requiredBufferSize =

Ivi_GetAttributeViString(vi, channel, attr, 0, 0, VI_NULL));

Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is readable. If not, the function returns an error.
- If the attribute cache value is currently valid, the read callback for the attribute is VI_NULL, or the IVI_ATTR_SIMULATE attribute is enabled and the IVI_ATTR_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function returns the cache value.
- 3. If the IVI_VAL_WAIT_FOR_OPC_BEFORE_READS flag is set for

the attribute, the function invokes the operation complete (OPC) callback you provide for the session.

- 4. The function invokes the read callback for the attribute. Typically, the callback performs instrument I/O to obtain a new value. The IVI engine stores the new value in the cache.
- 5. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

bufferSize

Vilnt32 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

that 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. 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 ir the Buffer

1

attributeValue

Size parameter. If the current value of the attribute, including the terminating NUL byte, contains more bytes that 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. If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can bring up a list of them on this control by pressing <ENTER>. Select a

value by doubleclicking on it or by highlightinç it and then pressing <ENTER>.

Return Value

Contains the status code that the function call returns.

If the function succeeds and the buffer you pass is large enough to hold the entire value, the function returns 0.

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 the function fails for some other reason, it returns a negative error code. For more information on error codes, refer to the Status return value control in one of the other function panels.

Related Topic

IVI Status Codes

Ivi_GetAttrMinMaxViInt32

Usage

ViStatus Ivi_GetAttrMinMaxViInt32(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32* minimum, ViInt32* maximum, ViBoolean* hasMinimum, ViBoolean* hasMaximum);

Purpose

If the range table for the attribute contains a meaningful minimum value, the function returns the minimum value in this parameter.

For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the minimum value is the lowest discreteOrMinValue in the table.

For tables of type IVI_VAL_COERCED, the minimum value is the lowest coercedValue in the table.

If you are not interested in this value, you can pass VI_NULL.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the
specific driver include file uses the
same constant name that appears in the
instrument class include file, except that
the specific instrument prefix replaces
the class prefix. For example, the DMM
class include file, ividmm.h, defines
IVIDMM_ATTR_RANGE, and fl45.h
defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE.

For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

Vilnt32* If the range table for the attribute contains a meaningful minimum value, the function returns the minimum value

minimum

		in this parameter.
		For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the minimum value is the lowest discreteOrMinValue in the table.
		For tables of type IVI_VAL_COERCED, the minimum value is the lowest coercedValue in the table.
		If you are not interested in this value, you can pass VI_NULL.
maximum	Vilnt32*	If the range table for the attribute contains a meaningful maximum value, the function returns the maximum value in this parameter.
		For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the maximum value is the highest discreteOrMinValue in the table.
		For tables of type IVI_VAL_RANGED, the maximum value is the highest maxValue in the table.
		For tables of type IVI_VAL_COERCED, the maximum value is the highest coercedValue in the table.
		If you are not interested in this value, you can pass VI_NULL.
hasMinimum	ViBoolean*	Returns VI_TRUE (1) if the range table for the attribute indicates that it contains a meaningful minimum value. Otherwise, returns VI_FALSE (0).
		If you are not interested in this value, you can pass VI_NULL.
		If you are not interested in this value,

you can pass VI_NULL.

hasMaximumViBoolean*Returns VI_TRUE (1) if the range table
for the attribute indicates that it contains
a meaningful maximum value.
Otherwise, returns VI_FALSE (0).
If you are not interested in this value,

you can pass VI_NULL.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttrMinMaxViInt64

Usage

ViStatus Ivi_GetAttrMinMaxViInt64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt64* minimum, ViInt64* maximum, ViBoolean* hasMinimum, ViBoolean* hasMaximum);

Purpose

If the range table for the attribute contains a meaningful minimum value, the function returns the minimum value in this parameter.

For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the minimum value is the lowest discreteOrMinValue in the table.

For tables of type IVI_VAL_COERCED, the minimum value is the lowest coercedValue in the table.

If you are not interested in this value, you can pass VI_NULL.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines

IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the followinc constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define HP34401A_ATTR_TRIGGER_TYPE \

		(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)
minimum	Vilnt64*	If the range table for the attribute contains a meaningful minimum value, the function returns the minimum value in this parameter.
		For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the minimum value is the lowest discreteOrMinValue in the table.
		For tables of type IVI_VAL_COERCED, the minimum value is the lowest coercedValue in the table.
		If you are not interested in this value, you can pass VI_NULL.
maximum	Vilnt64*	If the range table for the attribute contains a meaningful maximum value, the function returns the maximum value in this parameter.
		For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the maximum value is the highest discreteOrMinValue in the table.
		For tables of type IVI_VAL_RANGED, the maximum value is the highest maxValue in the table.
		For tables of type IVI_VAL_COERCED, the maximum value is the highest coercedValue in the table.
		If you are not interested in this value, you can pass VI_NULL.
hasMinimum	ViBoolean*	Returns VI_TRUE (1) if the range table for the attribute indicates that it contains

		a meaningful minimum value. Otherwise, returns VI_FALSE (0).
	If you are not interested in this value, you can pass VI_NULL.	
	If you are not interested in this value, you can pass VI_NULL.	
hasMaximum	ViBoolean*	Returns VI_TRUE (1) if the range table for the attribute indicates that it contains a meaningful maximum value. Otherwise, returns VI_FALSE (0).
		If you are not interested in this value, you can pass VI_NULL.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttrMinMaxViReal64

Usage

ViStatus Ivi_GetAttrMinMaxViReal64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViReal64* minimum, ViReal64* maximum, ViBoolean* hasMinimum, ViBoolean* hasMaximum);

Purpose

This function returns the minimum and maximum values that an instrument implements for a ViReal64 attribute on a specific repeated capability. The values represent the minimum and maximum values the driver or instrument actually uses rather than the possible values you can pass to Ivi_SetAttributeViReal64. In particular, for a coerced range table, the function uses the coercedValue fields.

The function calls Ivi_GetAttrRangeTable to obtain the range table for the attribute. If the attribute has no range table or the table is invalid, the function returns an error.

The hasMin and hasMax fields in the range table indicate whether, as a whole, the table contains a meaningful minimum value and a meaningful maximum value. The function returns these indicators.

If the hasMin field in the table is non-zero, the function searches the table for the minimum value. For discrete and ranged tables, the function examines the discreteOrMinValue field in each entry. For coerced tables, the function examines the coercedValue field.

If the hasMax field in the table is non-zero, the function searches the table for the maximum value. For discrete tables, the function examines the discreteOrMinValue field in each entry. For ranged tables, the function examines the maxValue field. For coerced tables, the function examines the coercedValue field.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the followinc constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

minimum

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

ViReal64* If the range table for the attribute contains a meaningful minimum value, the function returns the minimum value

		in this parameter.
		For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the minimum value is the lowest discreteOrMinValue in the table.
		For tables of type IVI_VAL_COERCED, the minimum value is the lowest coercedValue in the table.
		If you are not interested in this value, you can pass VI_NULL.
maximum	ViReal64*	If the range table for the attribute contains a meaningful maximum value, the function returns the maximum value in this parameter.
		For tables of type IVI_VAL_DISCRETE or IVI_VAL_RANGED, the maximum value is the highest discreteOrMinValue in the table.
		For tables of type IVI_VAL_RANGED, the maximum value is the highest maxValue in the table.
		For tables of type IVI_VAL_COERCED, the maximum value is the highest coercedValue in the table.
		If you are not interested in this value, you can pass VI_NULL.
hasMinimum	ViBoolean*	Returns VI_TRUE (1) if the range table for the attribute indicates that it contains a meaningful minimum value. Otherwise, returns VI_FALSE (0).
		If you are not interested in this value, you can pass VI_NULL.
		If you are not interested in this value,

you can pass VI_NULL.

hasMaximumViBoolean*Returns VI_TRUE (1) if the range table
for the attribute indicates that it contains
a meaningful maximum value.
Otherwise, returns VI_FALSE (0).
If you are not interested in this value,

you can pass VI_NULL.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_GetAttrRangeTable

Usage

ViInt32 = Ivi_GetAttrRangeTable(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, IviRangeTablePtr* rangeTable);

Purpose

This function returns a pointer to the range table for an attribute. If you call Ivi_SetAttrRangeTableCallback to install a range table callback function for the attribute, Ivi_GetAttrRangeTable invokes your range table callback with the vi, AttributeID, and RepeatedCapability parameters. Otherwise, Ivi_GetAttrRangeTable returns the address of the range table you specify for the attribute when you call Ivi_AddAttributeViInt32, Ivi AddAttributeViReal64, or Ivi SetStoredRangeTablePtr.

To bypass the range table callback and always return the range table you store for the attribute, call Ivi_GetStoredRangeTablePtr.

If you install your own check callback function in addition to either a range table or a range table callback, call this function from the check callback to obtain a pointer to the range table.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you ol from Ivi_SpecificDriverNew. The hidentifies a particular IVI session.
repeatedCapability	v ViChar[]	If the attribute is repeated capabi based, specify a particular repeated capability. If the attribute you spe not repeated capability-based, pa VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific inst driver defines constant names fo the user-accessible attributes tha to the driver. This includes attribu that the IVI engine defines, attributes that the instrument class defines, attributes that are specific to the particular instrument. Each define constant name begins with PREFIX_ATTR_, where PREFIX is specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses th same constant name that appear ivi.h, except that the specific insti prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke include file, fl45.h, defines the foll constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribu specific driver include file uses th same constant name that appear instrument class include file, exce the specific instrument prefix repl the class prefix. For example, the class include file, ividmm.h, define IVIDMM_ATTR_RANGE, and fl4 defines the following constant na

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attri the specific driver include file def constant name and assigns a val is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_B. For example, fl45.h defines the fc constant name:

#define

FL45_ATTR_HOLD_THRESHO (IVI_SPECIFIC_PUBLIC_ATTR_ + 3L)

For each attribute that is private t instrument driver, the instrument source file defines a constant nai assigns a value that is an offset f IVI_SPECIFIC_PRIVATE_ATTR_E For example, hp34401a.c defines following constant name:

#define

HP34401A_ATTR_TRIGGER_TY (IVI_SPECIFIC_PRIVATE_ATTF + 1L)

rangeTable

IviRangeTablePtr* Th

This parameter returns the addre the range table for the attribute.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes
Ivi_GetChannelIndex

Usage

ViStatus Ivi_GetChannelIndex(ViSession vi, ViChar ChannelName[], ViInt32* Index);

This function obtains the 1-based index of a channel name in the internal channel table for an IVI session.

If you pass VI_NULL or an empty string for the Channel Name parameter, this function sets the Index output parameter to 1.

If you pass a specific driver channel string for the Channel Name parameter, this function sets the Index output parameter to the 1-based index of the channel string in the internal channel table.

If you pass a virtual channel name that the end-user specifies in the configuration store, this function finds the specific driver channel string that the end-user assigns to the virtual channel name. The function then sets the Index output parameter to the 1-based index of the specific driver channel string in the internal channel table.

If you pass any other value for the Channel Name parameter, this function sets the Index output parameter to 0 and returns an error code.

Name	Туре	Description		
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.		
ChannelName	ViChar[]	Pass the channel name for which you want to obtain the index.		
		You can pass one of the following types of values:		
		 VI_NULL or an empty string. A specific driver channel string, which is one that the specific instrument driver specifies as valid using Ivi_BuildChannelTable or Ivi_AddToChannelTable. A virtual channel name that the end- user specifies in the configuration store. 		
Index	Vilnt32*	A 1-based index into the channel table.		
		If you pass an invalid value for the Channel Name parameter, the function sets this output parameter to 0 and returns an error code.		

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetConfigStoreHandle

Usage

ViStatus Ivi_GetConfigStoreHandle(IviConfigStoreHandle* handle);

Returns a handle to the IVI Configuration Server instance currently in use by the IVI engine. You can use this handle with the IVI Foundationdefined Configuration Server C API to add or delete runtime configuration elements.



Note Any changes you make to the Configuration Server through this handle are not saved, unless you explicitly save them through the Configuration Server C API.

If the Configuration Store file is modified by another process or IVI Configuration Server instance, the handle returned by this function becomes invalid, and you should not use the handle. Exception: This does not apply if you specified the handle using the Ivi_AttachToConfigStoreHandle.

Name	Туре	Description
handle	IviConfigStoreHandle*	Returns a handle to the configuration server instance currently in use by the IVI engine.
		You can use this handle with the IVI Foundation-defined Configuration Server

C API to add or delete runtime configuration elements.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

lvi_GetErrorInfo

Usage

ViStatus Ivi_GetErrorInfo(ViSession vi, ViStatus* primaryError, ViStatus* secondaryError, ViChar errorElaboration[]);

This function retrieves and then clears the error information for an IVI session or for the current execution thread. If you specify a valid IVI session for the vi parameter, this function retrieves and then clears the error information for the session. If you pass VI_NULL for the vi parameter, this function retrieves and then clears the error information for thread.

Instrument drivers export this function to the end-user through the PREFIX_GetErrorInfo function. Normally, the error information describes the first error that occurred since the end-user last called PREFIX_GetErrorInfo or PREFIX_ClearErrorInfo.

The error information includes a primary error code, secondary code error, and an error elaboration string. For a particular session, this information is the same as the values held in the following attributes:

- IVI_ATTR_PRIMARY_ERROR or PREFIX_ATTR_PRIMARY_ERROR
- IVI_ATTR_SECONDARY_ERROR or PREFIX_ATTR_SECONDARY_ERROR
- IVI_ATTR_ERROR_ELABORATION or PREFIX_ATTR_ERROR_ELABORATION

The IVI engine also maintains this error information separately for each thread. This is useful if you do not have a session handle to pass to Ivi_SetErrorInfo or Ivi_GetErrorInfo, which occurs when a call to Ivi_SpecificDriverNew fails.

Normally, it is the responsibility of the end-user to decide when to clear the error information by calling PREFIX_GetErrorInfo or PREFIX_ClearErrorInfo. If an instrument driver calls Ivi_GetErrorInfo, it must restore the error information by calling Ivi_SetErrorInfo, possibly adding a secondary error code or elaboration string.

You can call Ivi_GetErrorMessage to obtain a text description of the primary or secondary error value.

Name	Туре	Description
vi	ViSession	To obtain the error information for a particular IVI session, pass the ViSession handle that you obtain from Ivi_SpecificDriverNew.
		To obtain the error information for the current thread, pass VI_NULL.
primaryError	ViStatus*	The primary error code for the session or execution thread.
		A value of VI_SUCCESS (0) indicates that no error occurred. A positive value indicates a warning. A negative value indicates an error.
		You can call Ivi_GetErrorMessage or PREFIX_error_message to get a text description of the value.
		If you are not interested in this value, you can pass VI_NULL.
secondaryError	ViStatus*	The secondary error code for the session or execution thread. If the primary error code is non-zero, the secondary error code can further describe the error or warning condition.
		A value of VI_SUCCESS (0) indicates no further description.
		You can call Ivi_GetErrorMessage or PREFIX_error_message to get a text description of the value.
		If you are not interested in this value, you can pass VI_NULL.
errorElaboration	ViChar[]	

The error elaboration string for the session or execution thread. If the primary error code is non-zero, the elaboration string can further describe the error or warning condition.

If you are not interested in this value, you can pass VI_NULL. Otherwise, you must pass a ViChar array with at least IVI_MAX_MESSAGE_BUF_SIZE (256) bytes.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetErrorMessage

Usage

ViStatus Ivi_GetErrorMessage(ViStatus statusCode, ViChar statusMessage[]);

This function converts an IVI or VISA status code into a meaningful message string. For all other values, it reports the "Unknown status value" message and returns the VI_WARN_UNKNOWN_STATUS warning code.

If you have a table of error codes and messages that are specific to the instrument driver, call Ivi_GetSpecificDriverStatusDesc instead.

Parameters		
Name	Туре	Description
statusCode	ViStatus	A status code that an IVI function, a VISA function, or an instrument driver function returns.
statusMessage	ViChar[]	Returns a meaningful message string for an IVI or VISA status code. For other status codes, returns "Unknown status value".
		You must pass a ViChar array that contains at least IVI_MAX_MESSAGE_BUF_SIZE (256) bytes.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetInfoFromResourceName

Usage

ViStatus Ivi_GetInfoFromResourceName(ViRsrc resourceName, ViString optionString, ViChar newResourceName[], ViChar newOptionString[], ViBoolean* isLogicalName);

This function returns the appropriate resource name and option string that Ivi_SpecificDriverNew function uses to create a new IVI session. The Resource Name can be an actual resource descriptor or a logical name or driver session name that the user configures with the IVI Configuration utility.

If the string that the user passes in the Resource Name parameter is an actual resource descriptor, this function returns the original resource descriptor and option string in the New Resource Name and New Option String parameters and returns VI_FALSE in the Is Logical Name parameter.

If the string that the user passes in the Resource Name parameter is a logical name or driver session name, this function returns strings in the New Resource Name and New Option String parameters based on the settings of the logical name or virtual instrument in the IVI Configuration utility. The function return VI_TRUE in the Is Logical Name parameter.

Example:

Ivi_GetInfoFromResourceName ("GPIB0::2::INSTR",

```
"Simulate=1",newRsrcString, newOptionString, &isLogicalName);
```

newRsrcString and newOptionString contain the same values you pass to the function, and isLogicalName is VI_FALSE.

Ivi_GetInfoFromResourceName ("SampleDMM", "",

newRsrcString, newOptionString, &isLogicalName);

newRsrcString and newOptionString now contain the resource descriptor from the IVI configuration and the option that tells the engine through the Ivi_SpecificDriverNew function that the initial session setup comes from the IVI configuration. isLogicalName is VI_TRUE.

Name	Туре	Description
resourceName	ViRsrc	This parameter specifies the resource name of the specific instrument.
		The user can either pass an actual resource descriptor, such as "GPIB0::2::INSTR", or a logical name or driver session name that they configure with the IVI Configuration utility, such as "SampleDMM" or "MyFluke45."
optionString	ViString	This parameter is the option string that the user passes to the InitWithOptions function of the instrument driver.
newResourceName	ViChar[]	If the string that the user passes for the Resource Name parameter is an actual resource descriptor, this function returns the original value of the Resource Name parameter in this parameter.
		If the string that the user passes for the Resource Name parameter is a logical name or driver session name, then this parameter contains a resource descriptor that identifies the physical device based on the configuration of the logical name or virtual instrument in the IVI Configuration utility.
newOptionString	ViChar[]	If the string that the user passes for the Resource Name parameter is an actual resource descriptor, this

		function returns the original value of the Option String parameter in this parameter.
		If the string that the user passes for the Resource Name parameter is a logical name or driver session name, then this parameter returns a new option string. The new option string identifies the logical name or virtual instrument that the Ivi_SpecificDriverNew function uses to configure the initial configuration of the new IVI session.
isLogicalName	ViBoolean*	Returns VI_FALSE if the user passes an actual resource name for the Resource Name parameter.
		Returns VI_TRUE if the user passes a logical name or driver session name for the Resource Name parameter.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetInvalidationList

Usage

ViStatus Ivi_GetInvalidationList(ViSession vi, ViAttr attributeID, IviInvalEntry** invalidationList, ViInt32* numberOfEntries);

This function returns a list of all the invalidation dependency relationships for the session. The specific driver creates the dependency relationships using Ivi_AddAttributeInvalidation.

The function dynamically allocates an array of IviInvalEntry structures and returns a pointer to it. The last entry in the array is a termination entry that has IVI_ATTR_NONE (-1) in the attribute field. It also returns the number of items in the array, excluding the termination entry. When you are done with the list, you must free it by calling Ivi_DisposeInvalidationList.

You can pass VI_NULL for the Invalidation List parameter, in which case the function just returns the number of dependency relationships.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrumer driver defines constant names for all of the user-accessible attributes that appl to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrumer prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces

the class prefix. For example, the DMN class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the followir constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASI + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name an assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BAS + 1L)

invalidationList IviInvalEntry** Returns the pointer to an array that contains all the invalidation dependenc relationships for the session. The function dynamically allocates an

array of IviInvalEntry structures and returns a pointer to it in this parameter. The last entry in the array is a termination entry that has

	IVI_ATTR_NONE (-1) in the attribute field. When you are done with the list, you must free it by calling Ivi_DisposeInvalidationList.
	You can pass VI_NULL for this parameter, in which case the function just returns the number of dependency relationships.
numberOfEntries Vilnt32*	Returns the number of entries in the invalidation list, excluding the termination entry.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetLogicalNamesList

Usage

ViStatus Ivi_GetLogicalNamesList(IviLogicalNameEntry** logicalNamesList, ViInt32* numberOfEntries);

This function returns a list of the logical names that the IVI engine currently recognizes. You can define logical names in the IVI configuration file.

You pass logical names to class driver initialization functions to identify the physical device and specific driver module you want to use in a session.

The ivi.h include file defines the structure of an entry in the list as follows.

typedef struct

{ViString logicalName; ViBoolean fromFile; } IviLogicalNameEntry;

The fromFile field is always set to VI_TRUE.

The function dynamically allocates an array of IviLogicalNameEntry structures and returns a pointer to it. The logical names you define at runtime appear before the logical names from the configuration file. The last entry in the array is a termination entry that has VI_NULL in the logicalName field. The function also returns the number of logical names in the list, excluding the termination entry. When you are done with the list, you must free it by calling Ivi_DisposeLogicalNamesList.

Call Ivi_GetNthLogicalName to extract the data from an entry in the list. Do not change the values of any of the entries in the list.

You can pass VI_NULL for the Logical Names List parameter, in which case the function just returns the number of logical names.

Name	Туре	Description
logicalNamesList	IviLogicalNameEntry**	Returns the pointer to an array of the logical names that the IVI engine currently recognizes.
numberOfEntries	Vilnt32*	Returns the number of entries in the logical names list, excluding the termination entry.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetNextCoercionInfo

Usage

ViStatus Ivi_GetNextCoercionInfo(ViSession vi, ViAttr* attributeID, ViConstString* attributeName, ViConstString* repeatedCapabilityname, IviValueType* attributeDataType, ViReal64* desiredValue, ViReal64* coercedValue);

This function obtains information regarding the oldest instance in which the IVI engine coerced an attribute value you specified to another value. It then deletes that information.

If you enable the IVI_ATTR_RECORD_COERCIONS attribute for the session, the IVI engine keeps a list of all coercions it makes on values you pass to an Ivi_SetAttribute function for a ViInt32 or ViReal64 attribute. You can use this function to retrieve information from that list. Each time you call this function, it extracts and deletes the oldest coercion record for the session.

When no coercion records remain for the session, the function returns IVI_ATTR_NONE (-1) in the Attribute ID parameter and VI_NULL in the Attribute Name parameter.

The function returns all numeric values as ViReal64 values, even for Vilnt32 attributes.

You can pass VI_NULL for any of the output parameters, except that you cannot pass VI_NULL for both the Attribute ID and Attribute Name parameters in one call.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular I\ session.
attributeID	ViAttr*	Returns the ID of the attribute which the value coercion occurred. No more coercion records exist for session, this parameter returns IVI_ATTR_NONE (-1).
		If you are not interested in this value, you can pass VI_NULL as long as you do not also pas VI_NULL for the Attribute Nan parameter.
attributeName	ViConstString*	Returns a pointer to the name the attribute for which the valu- coercion occurred. If no more coercion records exist for session, this parameter returns VI_NULL.
		Do not modify the contents of the name.
		If you are not interested in this value, you can pass VI_NULL as long as you do not also pas VI_NULL for the Attribute ID parameter.
repeatedCapabilityname	ViConstString*	If the attribute is repeated capability-based, this paramete

		returns a pointer to the name of the repeated capability on whice the value coercion occurred. If the attribute is not repeated capability-based, this parameter returns a pointer to an empty string.
		Do not modify the contents of the repeated capability string.
		If you are not interested in this value, you can pass VI_NULL
attributeDataType	IviValueType*	Returns the data type of the attribute.
		Values:
		(1) IVI_VAL_INT32 Vilnt32
		(4) IVI_VAL_REAL64 ViReal
		If you are not interested in this value, you can pass VI_NULL
desiredValue	ViReal64*	Returns the value to which you attempted to set the attribute. The function always returns the value as a ViReal64 value, eve if the data type of the attribute Vilnt32.
		If you are not interested in this value, you can pass VI_NULL
coercedValue	ViReal64*	Returns the value to which the IVI engine actually set the attribute. The function always returns the value as a ViReal6 value, even if the data type of the attribute is ViInt32.
If you are not interested in this value, you can pass VI_NULL.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetNextCoercionString

Usage

ViStatus Ivi_GetNextCoercionString(ViSession vi, ViInt32 bufferSize, ViChar coercionString[]);

This function obtains information regarding the oldest instance in which the IVI engine coerced an attribute value you specified to another value. This information is in a string format. It then deletes that information.

If you enable the IVI_ATTR_RECORD_COERCIONS attribute for the session, the IVI engine keeps a list of all coercions it makes on values you pass to an Ivi_SetAttribute function for a ViInt32 or ViReal64 attribute. You can use this function to retrieve information from that list. Each time you call this function, it extracts and deletes the oldest coercion record for the session.

When no coercion records remain for the session, the function returns an empty string ("") in the Coercion String parameter.

The function returns the string containing the coercion information. You must provide a ViChar array to serve as a buffer for the string. You pass the number of bytes in the buffer as the Buffer Size parameter. If the current size of the coercion string, 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 string. 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 the function to fill in the buffer regardless of the number of bytes in the string, pass a negative number for the Buffer Size parameter. 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 Coercion String buffer.

Remember that the checkErr and viCheckErr macros ignore positive return values. If you use one of these macros around a call to this function, you lose the required buffer size when the function returns it. To retain this information, declare a separate local variable to store the required buffer size, and use the macro around the assignment of the return value to the local variable. The following is an example:

```
ViStatus error = VI_SUCCESS;ViInt32 requiredBufferSize;
```

```
checkErr( requiredBufferSize =
Ivi_GetNextCoercionString(vi, 0, VI_NULL));
```

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
bufferSize	Vilnt32	Pass the number of bytes in the ViChar array you specify for the Coercion String parameter.
		If the current coercion 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 string. 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 coercion string to the buffer regardless of the number of bytes in the string.
		If you pass 0, you can pass VI_NULL for the Coercion String buffer parameter.
coercionString	ViChar[]	The buffer in which the function returns the description of the oldest value coercion of the instrument session. The buffer must be of type ViChar and have at least as many bytes as indicated in the Buffer Size parameter.
		If the current coercion 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 string. For example, if the string 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.

When no coercion records remain for the session, the function returns an empty string ("") in the Coercion String parameter.

Contains the status code that the function call returns.

If the function succeeds and the buffer you pass is large enough to hold the entire value, the function returns 0.

If the current length of the coercion string, 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 string. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If the function fails for some other reason, it returns a negative error code. For more information on error codes, refer to the Status return value control in one of the other function panels.

Related Topic

Ivi_GetNextInterchangeCheckString

Usage

ViStatus Ivi_GetNextInterchangeCheckString(ViSession vi, ViInt32 bufferSize, ViChar interchangeWarning[]);

This function returns the interchangeability warnings associated with the IVI session. Interchangeability warnings indicate that using your application with a different instrument might cause different behavior.

The specific driver performs interchangeability checking if the IVI_ATTR_INTERCHANGE_CHECK attribute is set to VI_TRUE. Each time you call this function, it extracts and deletes the oldest interchangeability warning information for the session.

If the next interchangeability warning string, including the terminating NUL byte, contains more bytes than you indicate in bufSize parameter, the function copies bufSize - 1 bytes into the interchangeWarning buffer, places an ASCII NUL byte at the end of the interchangeWarning 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 interchange warning buffer and returns 7.

If you want the function to fill in the buffer regardless of the number of bytes in the string, pass a negative number for the bufSize parameter. If you want to call this function just to get the required buffer size, you can pass 0 for the bufSize and VI_NULL for the interchangeWarning buffer.

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

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
bufferSize	Vilnt32	Pass the number of bytes in the ViChar array you specify for the interchange warning parameter.
		If the current interchange warning, including the terminating NUL byte, contains more bytes that 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 string. 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 interchange warning to the buffer regardless of the number of bytes in the string.
		If you pass 0, you can pass VI_NULL for the Interchange Warning buffer parameter.
interchangeWarning	ViChar[]	The buffer in which the function returns the description of the oldest value coercion of the instrument session. The buffer must be of type ViChar and have at least as many bytes as indicated in the Buffer Size

parameter.

If the current coercion string, including the terminating NUL byte, contains more bytes that 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 string. For example, if the string 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.

When no interchange warnings remain for the session, the function returns an empty string ("") in the Coercion String parameter.

Contains the status code that the function call returns.

If the function succeeds and the buffer you pass is large enough to hold the entire value, the function returns 0.

If the current length of the interchange warning, 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 string. For example, if the value is "123456" and the Buffer Size is 4, the function places "123" into the buffer and returns 7.

If the function fails for some other reason, it returns a negative error code. For more information on error codes, refer to the Status return value control in one of the other function panels.

Related Topic

Ivi_GetNthAttribute

Usage

ViStatus Ivi_GetNthAttribute(ViSession vi, ViInt32 index, ViAttr* attributeID);

This function obtains the ID of the attribute that is at the index you specify in the IVI session's internal list of attributes. The index is 1-based.

If the index you specify is greater than the number of attributes, the function sets the Attribute ID parameter to IVI_ATTR_NONE (-1) and returns VI_SUCCESS.

Call Ivi_GetNumAttributes to determine the number of attributes in the internal list.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
index	Vilnt32	Specify a 1-based index into the IVI session's internal list of attributes.
attributeID ViAttr*		Returns the ID of the attribute at the selected index of the attribute list.
		If the index you specify is greater than the number of attributes, the function sets this parameter to IVI_ATTR_NONE (-1) and returns VI_SUCCESS.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetNthChannelString

Usage

ViStatus Ivi_GetNthChannelString(ViSession vi, ViInt32 index, ViConstString* channelString);

The function returns the channel string that is in the channel table at an index you specify. The specific instrument driver specifies the contents of the channel table using Ivi_BuildChannelTable and Ivi_AddToChannelTable, and the IVI engine maintains the table for the session.

If the index you specify is greater than the number of channel strings in the table, the function sets the Channel String parameter to VI_NULL and returns VI_SUCCESS.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
index	Vilnt32	A 1-based index into the channel table.
		If you pass an index that is greater than the number of strings in the table, the function sets the Channel String parameter to VI_NULL and returns VI_SUCCESS.
channelString	ViConstString*	Returns the channel string that is in the channel table at the index you specify.
		Do not modify the contents of the channel string.
		If the Index parameter is greater than the number of strings in the table, the function sets this parameter to VI_NULL and returns VI_SUCCESS.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetNthLogicalName

Usage

ViStatus Ivi_GetNthLogicalName(IviLogicalNameEntry* logicalNamesList, ViInt32 index, ViChar logicalNameBuffer[], ViInt32 bufferSize, ViBoolean* fromFile);

This function extracts the data from an entry in a logical names list you obtain from Ivi_GetLogicalNamesList. You specify the entry with a 1-based index.

If the index you specify is greater than the number of logical names, the function places an ASCII NUL byte at the beginning of the Logical Name Buffer parameter and returns VI_SUCCESS.

Name	Туре	Description
logicalNamesList	IviLogicalNameEntry*	Specify the pointer to the logical names list you obtain from Ivi_GetLogicalNamesList.
index	Vilnt32	Specify the 1-based index of the logical name list entry from which you want to extract data.
logicalNameBuffer	ViChar[]	The buffer in which the function returns the logical name. The buffer must be a ViChar array that contains at least as many bytes as you specify in the Buffer Size parameter.
		If the index you specify is greater than the number of logical names, the function places an ASCII NUL byte at the beginning of the buffer and returns VI_SUCCESS.
bufferSize	Vilnt32	Specify the number of bytes in the Logical Name Buffer parameter.
fromFile	ViBoolean*	This parameter always returns a value of VI_TRUE (1). You can pass VI_NULL to this parameter.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetNthRepCapString

Usage

ViStatus Ivi_GetNthRepCapString(ViSession vi, ViChar repeatedCapabilityName[], ViInt32 Index, ViConstString* repeatedCapabilityIdentifier);

The function returns the repeated capability string that is in the repeated capability table at an index you specify. The specific instrument driver specifies the contents of the table using Ivi_BuildRepCapTable and Ivi_AddToRepCapTable.

If the index you specify is greater than the number of strings in the table, the function sets the Repeated Capability Identifier parameter to VI_NULL and returns VI_SUCCESS.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapabilityName	ViChar[]	Pass a string containing the name of the repeated capability on which to operate. For instance, if you are working with the table of channel names, pass in the string "Channel".
Index	Vilnt32	A 1-based index into the channel table.
		If you pass an index that is greater than the number of strings in the table, the function sets the Channel String parameter to VI_NULL and returns VI_SUCCESS.
repeatedCapabilityIdentifier	ViConstString*	Returns the repeated capability string that is in the repeated capability table at the index you specify.
		Do not modify the contents of the repeated capability string.
		If the Index parameter is

greater than the number of strings in the table, the function sets this parameter to VI_NULL and returns VI_SUCCESS.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetNumAttributes

Usage

ViStatus Ivi_GetNumAttributes(ViSession vi, ViInt32* numberOfAttributes);

This function obtains the total number of attributes in the IVI session you specify. This includes all attributes that the IVI engine and the driver create, regardless of whether the IVI_VAL_NOT_SUPPORTED flag for the attribute is set.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
numberOfAttributes	Vilnt32*	Returns the total number of attributes in the IVI session. This includes all attributes that the IVI engine and the driver create, regardless of whether the IVI_VAL_NOT_SUPPORTED flag for the attribute is set.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetRangeTableNumEntries

Usage

ViStatus Ivi_GetRangeTableNumEntries(IviRangeTablePtr rangeTable, ViInt32* numberOfEntries);

This function returns the number of entries in a range table, excluding the termination entry. If you pass VI_NULL for the Range Table parameter, the function returns 0 as the number of entries.

Parameters	S
------------	---

Name	Туре	Description
rangeTable	IviRangeTablePtr	Specify the address of the range table you want to examine.
		You can pass VI_NULL for this parameter.
numberOfEntries	Vilnt32*	Returns the total number of entries in the range table, excluding the termination entry. If you pass VI_NULL for the Range Table parameter, this parameter returns 0.
Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetRepCapIndex

Usage

ViStatus Ivi_GetRepCapIndex(ViSession vi, ViChar repeatedCapabilityName[], ViChar repeatedCapabilityIdentifier[], ViInt32* index);

This function obtains the 1-based index of a repeated capability in the internal repeated capability for an IVI session.

If you pass VI_NULL or an empty string for the Repeated Capability Identifier parameter, this function sets the Index output parameter to 1.

If you pass a specific driver identifier for the Repeated Capability Identifier parameter, this function sets the Index output parameter to the 1-based index of the identifier string in the internal repeated capability table.

If you pass a virtual repeated capability identifier, this function returns the one-based index of the specific driver string to which the virtual identifier coerces.

If you pass any other value for the Repeated Capability parameter, this function sets the Index output parameter to 0 and returns an error code.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapabilityName	ViChar[]	Pass a string containing the name of the repeated capability on which to operate. For instance, if you are working with the table of channel names, pass in the string "Channel".
repeatedCapabilityIdentifier	ViChar[]	Pass the repeated capability name for which you want to obtain the index.
		The parameter accepts VI_NULL, and empty string, a specific driver-defined repeated capability identifier, or a virtual repeated capability identifier.
index	Vilnt32*	A 1-based index into the repeated capability table.
		If you pass an invalid value for the Repeated Capability Identifier parameter, the function sets this output parameter to 0 and returns an error code.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetSpecificDriverStatusDesc

Usage

ViStatus Ivi_GetSpecificDriverStatusDesc(ViSession vi, ViStatus statusCode, IviStringValueTable additionalTableToSearch, ViChar statusMessage[]);

This function converts a status code that an instrument driver function returns into a meaningful message string. It interprets IVI and VISA status codes just as Ivi_GetErrorMessage does, but it also allows you to pass a table of error codes and messages that are specific to the instrument driver.

Use this function to implement the PREFIX_error_message function in the instrument driver.

If the function cannot find a description for the status code, it reports the "Unknown status value" message and returns the VI_WARN_UNKNOWN_STATUS warning code.

Par	ameters	
-----	---------	--

Name	Туре	Description
vi	ViSession	The ViSession handle tha obtain from Ivi_SpecificDriverNew.
		You can pass VI_NULL for parameter. This is useful Ivi_SpecificDriverNew fails
statusCode	ViStatus	A status code that an inst driver function returns.
additionalTableToSearch	IviStringValueTable	Specify a string/value tab contains status codes spe the instrument driver. Spe message string for each s code in the table. Termina table with an entry that ha VI_NULL in the string fiel
		The ivi.h include file define structure of a string/value entry as follows:
		typedef struct
		{
		ViInt32 value;ViString } IviStringValueEntry;
		If you pass VI_NULL for 1 parameter, the function be the same as Ivi_GetErrorMessage.
statusMessage	ViChar[]	Returns a meaningful me string for an IVI, VISA, or instrument driver status c the status code is unknov

returns "Unknown status

You must pass a ViChar a that contains at least IVI_MAX_MESSAGE_BU (256) bytes.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetStoredRangeTablePtr

Usage

ViStatus Ivi_GetStoredRangeTablePtr(ViSession vi, ViAttr attributeID, IviRangeTablePtr* rangeTable);

This function obtains the address of the range table you store for the attribute when you call Ivi_AddAttributeViInt32, Ivi_AddAttributeViReal64, or Ivi_SetStoredRangeTablePtr.

Unlike Ivi_GetAttrRangeTable, this function never invokes the range table callback.

Parameters

Name	Туре	Description		
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.		
attributeID	ViAttr	Pass the ID of an attribute for this parameter.		
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.		
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:		
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE		
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM		

class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

rangeTable IviRangeTablePtr* Returns the address of the range table that you store for this attribute. If you do not store a range table, this parameter returns VI_NULL.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetStringFromTable

Usage

ViStatus Ivi_GetStringFromTable(IviStringValueTable stringTable, ViInt32 value, ViString* string);

This function searches for a value in a string/value table and returns the string that corresponds to the value.

If the function cannot find the value in the table, it returns the IVI_ERROR_INVALID_VALUE error code.

Parameters

Name	Туре	Description
stringTable	IviStringValueTable	Specify the string/value table in which you want to find the string.
		The ivi.h include file defines the structure of a string/value table entry as follows:
		typedef struct
		{ ViInt32 value; ViString string; } IviStringValueEntry;
value	Vilnt32	Specify the value you want to find in the string/value table.
string	ViString*	If the function finds a string/value table entry that contains the value you specify, this parameter returns the address of the string in the entry.
		Do not modify the contents of the string.
		If the function does not find the value in the table, this parameter returns VI_NULL, and the function returns the IVI_ERROR_INVALID_VALUE error code.
		You can pass VI_NULL for this parameter.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetUserChannelName

Usage

ViStatus Ivi_GetUserChannelName(ViSession vi, ViChar channelString[], ViConstString* userChannelName);

This function finds the highest-level channel name that corresponds to the specific driver channel string you specify. It returns a pointer to the name in the User Channel Name output parameter.

If you specify a channel string that the end-user assigns to a virtual channel name in the ivi.ini configuration file, the function returns a pointer to the virtual channel name. If the end-user assigns the channel string to multiple virtual channel names, the function returns a pointer to the first virtual channel name it finds.

If no virtual channel names correspond to the channel string and the channel string is in the channel table that the specific instrument driver defines, the function returns a pointer to the channel string. If the channel string is not in the table, the function sets the User Channel Name output parameter to VI_NULL and returns an error code.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
channelString	ViChar[]	A specific driver channel string. The specific instrument driver specifies the valid channel strings using Ivi_BuildChannelTable and Ivi_AddToChannelTable.
userChannelName	ViConstString*	The highest-level channel name that corresponds to the specific driver channel string you pass in the Channel String parameter.
		Do not modify the contents of the channel name.
		This might be a virtual channel name, the channel string, or VI_NULL.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetValueFromTable

Usage

ViStatus Ivi_GetValueFromTable(IviStringValueTable stringTable, ViChar string[], ViInt32* value);

This function searches for a string in a string/value table and returns the value that corresponds to the string.

If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the strings in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the table entry, followed by a carriage return, linefeed, or ASCII NUL byte.

If the function cannot find the string in the table, it returns the IVI_ERROR_INVALID_VALUE error.

Parameters

Name	Туре	Description
stringTable IviStringValueTa	IviStringValueTable	Specify the string/value table in which you want to find the string.
		The ivi.h include file defines the structure of a string/value table entry as follows:
		typedef struct
		{ ViInt32 value; ViString string; } IviStringValueEntry;
string	ViChar[]	Specify the string you want to find in the string/value table.
		If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the strings in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the table entry, followed by a carriage return, newline, or ASCII NUL byte.
value	Vilnt32*	If the function finds a string/value table entry that contains the string you specify, this parameter returns the value in the entry.
		If the function does not find the value in the table, the function returns the IVI_ERROR_INVALID_VALUE error code.
		You can pass VI_NULL for this parameter.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt32EntryFromCmdValue

Usage

ViStatus Ivi_GetViInt32EntryFromCmdValue(ViInt32 commandValue, IviRangeTablePtr rangeTable, ViInt32* discreteOrMinValue, ViInt32* maxValue, ViInt32* coercedValue, ViInt32* tableIndex, ViString* commandString);

This function finds the first range table entry for which the cmdValue field is equal to the command value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt32 values.

Parameters

Name	Туре	Description
commandValue	Vilnt32	Specify the command value that you want to find in the range table.
rangeTable	IviRangeTablePtr	Specify the address of the range table in which to search for the command value.
discreteOrMinValue	Vilnt32*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt32*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	Vilnt32*	Specify the coerced value that you want to find in the range table.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

commandString

ViString*

If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.

Do not change the contents of the string.

You can pass VI_NULL for the parameter if you are not interested in this value.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt32EntryFromCoercedVal

Usage

ViStatus Ivi_GetViInt32EntryFromCoercedVal(ViInt32 coercedValue, IviRangeTablePtr rangeTable, ViInt32* discreteOrMinValue, ViInt32* maxValue, ViInt32* tableIndex, ViString* commandString, ViInt32* commandValue);

This function finds the first range table entry for which the coercedValue field is equal to the coerced value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue and maxValue fields as ViInt32 values.

Name	Туре	Description
coercedValue	Vilnt32	Specify the coerced value that you want to find in the range table.
rangeTable	lviRangeTablePtr	Specify the address of the range table in which to search for the coerced value.
discreteOrMinValue	Vilnt32*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt32*	If the search succeeds, this parameter returns the value of the maxValue field of the entry. You can pass VI_NULL for the parameter if you are not
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.

		Do not change the contents of the string.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	Specify the command value that you want to find in the range table.
Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt32EntryFromIndex

Usage

ViStatus Ivi_GetViInt32EntryFromIndex(ViInt32 tableIndex, IviRangeTablePtr rangeTable, ViInt32* discreteOrMinValue, ViInt32* maxValue, ViInt32* coercedValue, ViString* commandString, ViInt32* commandValue);

This function extracts the range table entry that is at the 0-based index you specify. The function returns the contents of the entry.

If you specify an index that is less than 0 or greater than or equal to the number of entries in the table, the function returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt32 values.

Parameters		
Name	Туре	Description
tableIndex	Vilnt32	Specify the 0-based index of the range table entry you want to extract.
rangeTable	IviRangeTablePtr	Specify the address of the range table from which to extract the entry at the index you specify.
discreteOrMinValue	Vilnt32*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt32*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	Vilnt32*	Specify the coerced value that you want to find in the range table.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry. Do not change the contents of the string.

		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	Specify the command value that you want to find in the range table.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt32EntryFromString

Usage

ViStatus Ivi_GetViInt32EntryFromString(ViChar commandString[], IviRangeTablePtr rangeTable, ViInt32* discreteOrMinValue, ViInt32* maxValue, ViInt32* coercedValue, ViInt32* tableIndex, ViInt32* commandValue);

This function finds the first range table entry for which the cmdString field matches the string you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function compares strings in a case-sensitive manner.

If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the cmdString fields in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the cmdString field, followed by a carriage return, newline, or ASCII NUL byte.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt32 values.

Parameters

Name	Туре	Description
commandString	ViChar[]	Specify the command string that you want to find in the range table.
		If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the cmdString fields in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the cmdString field, followed by a carriage return, newline, or ASCII NUL byte.
rangeTable	IviRangeTablePtr	Specify the address of the range table in which to search for the command string.
discreteOrMinValue	Vilnt32*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt32*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not

		interested in this value.
coercedValue	Vilnt32*	Specify the coerced value that you want to find in the range table.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	Specify the command value that you want to find in the range table.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt32EntryFromValue

Usage

ViStatus Ivi_GetViInt32EntryFromValue(ViInt32 value, IviRangeTablePtr rangeTable, ViInt32* discreteOrMinValue, ViInt32* maxValue, ViInt32* coercedValue, ViInt32* tableIndex, ViString* commandString, ViInt32* commandValue);

This function finds the first range table entry that applies to the ViInt32 value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

If the range table type is IVI_VAL_DISCRETE, function searches for a match on the discreteOrMinValue field of each entry.

If the range table type is IVI_VAL_RANGED or IVI_VAL_COERCED, the function searches until the value you specify falls within the range between the discreteOrMinValue and maxValue fields of an entry. The value falls within the range if it is greater than or equal to the discreteOrMinValue and less than or equal to the maxValue.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt32 values.

Parameters

Name	Туре	Description
value	Vilnt32	Specify the value that you want to find in the range table.
rangeTable	IviRangeTablePtr	Specify the address of the range table in which to search for the value.
discreteOrMinValue	Vilnt32*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt32*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	Vilnt32*	If the search succeeds, this parameter returns the value of the coercedValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.

		You can pass VI_NULL for the parameter if you are not interested in this value.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.
		Do not change the contents of the string.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	If the search succeeds, this parameter returns the value of the cmdValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt64EntryFromCmdValue

Usage

ViStatus Ivi_GetViInt64EntryFromCmdValue(ViInt32 commandValue, IviRangeTablePtr rangeTable, ViInt64* discreteOrMinValue, ViInt64* maxValue, ViInt64* coercedValue, ViInt32* tableIndex, ViString* commandString);

This function finds the first range table entry for which the cmdValue field is equal to the command value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt64 values.

Parameters

Name	Туре	Description
commandValue	Vilnt32	Specify the command value that you want to find in the range table.
rangeTable	IviRangeTablePtr	Specify the address of the range table in which to search for the command value.
discreteOrMinValue	Vilnt64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry. You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	Vilnt64*	If the search succeeds, this parameter returns the value of the coercedValue field of the entry. You can pass VI_NULL for the parameter if you are not
tableIndex	Vilnt32*	Interested in this value. If the search succeeds, this parameter returns the 0-based index of the entry.

	You can pass VI_NULL for the parameter if you are not interested in this value.
ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.
	Do not change the contents of the string.
	You can pass VI_NULL for the parameter if you are not interested in this value.
	ViString*

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt64EntryFromCoercedVal

Usage

ViStatus Ivi_GetViInt64EntryFromCoercedVal(ViInt64 coercedValue, IviRangeTablePtr rangeTable, ViInt64* discreteOrMinValue, ViInt64* maxValue, ViInt32* tableIndex, ViString* commandString, ViInt32* commandValue);

This function finds the first range table entry for which the coercedValue field is equal to the coerced value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue and maxValue fields as ViInt64 values.

Name	Туре	Description
coercedValue	Vilnt64	Specify the coerced value that you want to find in the range table.
rangeTable	lviRangeTablePtr	Specify the address of the range table in which to search for the coerced value.
discreteOrMinValue	Vilnt64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry. You can pass VI_NULL for the parameter if you are not
		interested in this value.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.

Do not change the contents of the string.

You can pass VI_NULL for the parameter if you are not interested in this value.

commandValue

Vilnt32*

If the search succeeds, this parameter returns the value of the cmdValue field of the entry.

You can pass VI_NULL for the parameter if you are not interested in this value.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt64EntryFromIndex

Usage

ViStatus Ivi_GetViInt64EntryFromIndex(ViInt32 tableIndex, IviRangeTablePtr rangeTable, ViInt64* discreteOrMinValue, ViInt64* maxValue, ViInt64* coercedValue, ViString* commandString, ViInt32* commandValue);

This function extracts the range table entry that is at the 0-based index you specify. The function returns the contents of the entry.

If you specify an index that is less than 0 or greater than or equal to the number of entries in the table, the function returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt64 values.

Parameters		
Name	Туре	Description
tableIndex	Vilnt32	Specify the 0-based index of the range table entry you want to extract.
rangeTable	lviRangeTablePtr	Specify the address of the range table from which to extract the entry at the index you specify.
discreteOrMinValue	Vilnt64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	Vilnt64*	If the search succeeds, this parameter returns the value of the coercedValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in

		the cmdString field of the entry.
		Do not change the contents of the string.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	If the search succeeds, this parameter returns the value of the cmdValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViInt64EntryFromString

Usage

ViStatus Ivi_GetViInt64EntryFromString(ViChar commandString[], IviRangeTablePtr rangeTable, ViInt64* discreteOrMinValue, ViInt64* maxValue, ViInt64* coercedValue, ViInt32* tableIndex, ViInt32* commandValue);

This function finds the first range table entry for which the cmdString field matches the string you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function compares strings in a case-sensitive manner.

If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the cmdString fields in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the cmdString field, followed by a carriage return, newline, or ASCII NUL byte.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt64 values.

Parameters

Name	Туре	Description
commandString	ViChar[]	Specify the command string that you want to find in the range table.
		If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the cmdString fields in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the cmdString field, followed by a carriage return, newline, or ASCII NUL byte.
rangeTable	lviRangeTablePtr	Specify the address of the range table in which to search for the command string.
discreteOrMinValue Vilnt64*	Vilnt64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not

		interested in this value.
coercedValue	Vilnt64*	If the search succeeds, this parameter returns the value of the coercedValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	If the search succeeds, this parameter returns the value of the cmdValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic
Ivi_GetViInt64EntryFromValue

Usage

ViStatus Ivi_GetViInt64EntryFromValue(ViInt64 value, IviRangeTablePtr rangeTable, ViInt64* discreteOrMinValue, ViInt64* maxValue, ViInt64* coercedValue, ViInt32* tableIndex, ViString* commandString, ViInt32* commandValue);

This function finds the first range table entry that applies to the ViInt64 value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

If the range table type is IVI_VAL_DISCRETE, function searches for a match on the discreteOrMinValue field of each entry.

If the range table type is IVI_VAL_RANGED or IVI_VAL_COERCED, the function searches until the value you specify falls within the range between the discreteOrMinValue and maxValue fields of an entry. The value falls within the range if it is greater than or equal to the discreteOrMinValue and less than or equal to the maxValue.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViInt64 values.

Parameters

Name	Туре	Description
value	Vilnt64	Specify the value that you want to find in the range table.
rangeTable	IviRangeTablePtr	Specify the address of the range table in which to search for the value.
discreteOrMinValue	Vilnt64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	Vilnt64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	Vilnt64*	If the search succeeds, this parameter returns the value of the coercedValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.

		You can pass VI_NULL for the parameter if you are not interested in this value.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.
		Do not change the contents of the string.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	If the search succeeds, this parameter returns the value of the cmdValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViReal64EntryFromCmdValue

Usage

ViStatus Ivi_GetViReal64EntryFromCmdValue(ViInt32 commandValue, IviRangeTablePtr rangeTable, ViReal64* discreteOrMinValue, ViReal64* maxValue, ViReal64* coercedValue, ViInt32* tableIndex, ViString* commandString);

This function finds the first range table entry for which the cmdValue field is equal to the command value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as VilReal64 values.

Parameters

Name	Туре	Description
commandValue	Vilnt32	Specify the command value that you want to find in the range table.
rangeTable	IviRangeTablePtr	Specify the address of the range table in which to search for the command value.
discreteOrMinValue	ViReal64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	ViReal64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	ViReal64*	Specify the coerced value that you want to find in the range table.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

commandString

ViString*

If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.

Do not change the contents of the string.

You can pass VI_NULL for the parameter if you are not interested in this value.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViReal64EntryFromCoercedVal

Usage

ViStatus Ivi_GetViReal64EntryFromCoercedVal(ViReal64 coercedValue, IviRangeTablePtr rangeTable, ViReal64* discreteOrMinValue, ViReal64* maxValue, ViInt32* tableIndex, ViString* commandString, ViInt32* commandValue);

This function finds the first range table entry for which the coercedValue field is equal to the coerced value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function performs all ViReal64 comparisons using a comparison precision of 14 decimal digits.

The function returns the discreteOrMinValue and maxValue fields as ViReal64 values.

Name	Туре	Description
coercedValue	ViReal64	Specify the coerced value that you want to find in the range table.
rangeTable	lviRangeTablePtr	Specify the address of the range table in which to search for the coerced value.
discreteOrMinValue	ViReal64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	ViReal64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry. You can pass VI_NULL for the parameter if you are not interested in this value.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry.

		Do not change the contents of the string.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	Specify the command value that you want to find in the range table.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViReal64EntryFromIndex

Usage

ViStatus Ivi_GetViReal64EntryFromIndex(ViInt32 tableIndex, IviRangeTablePtr rangeTable, ViReal64* discreteOrMinValue, ViReal64* maxValue, ViReal64* coercedValue, ViString* commandString, ViInt32* commandValue);

This function extracts the range table entry that is at the 0-based index you specify. The function returns the contents of the entry.

If you specify an index that is less than 0 or greater than or equal to the number of entries in the table, the function returns an IVI_ERROR_INVALID_VALUE error.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViReal64 values.

Parameters		
Name	Туре	Description
tableIndex	Vilnt32	Specify the 0-based index of the range table entry you want to extract.
rangeTable	IviRangeTablePtr	Specify the address of the range table from which to extract the entry at the index you specify.
discreteOrMinValue	ViReal64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	ViReal64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	ViReal64*	Specify the coerced value that you want to find in the range table.
commandString	ViString*	If the search succeeds, this parameter returns the pointer in the cmdString field of the entry. Do not change the contents of the string.

		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	Specify the command value that you want to find in the range table.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViReal64EntryFromString

Usage

ViStatus Ivi_GetViReal64EntryFromString(ViChar commandString[], IviRangeTablePtr rangeTable, ViReal64* discreteOrMinValue, ViReal64* maxValue, ViReal64* coercedValue, ViInt32* tableIndex, ViInt32* commandValue);

This function finds the first range table entry for which the cmdString field matches the string you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

The function compares strings in a case-sensitive manner.

If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the cmdString fields in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the cmdString field, followed by a carriage return, newline, or ASCII NUL byte.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViReal64 values.

Parameters

Name	Туре	Description
commandString	ViChar[]	Specify the command string that you want to find in the range table.
		If the string you specify terminates with a carriage return ('\r') or newline ('\n') character, the cmdString fields in the table do not have to contain the termination character. The function considers the strings to match if the string you specify begins with the string in the cmdString field, followed by a carriage return, newline, or ASCII NUL byte.
rangeTable	lviRangeTablePtr	Specify the address of the range table in which to search for the command string.
discreteOrMinValue	ViReal64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	ViReal64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not

		interested in this value.
coercedValue	ViReal64*	Specify the coerced value that you want to find in the range table.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandValue	Vilnt32*	Specify the command value that you want to find in the range table.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViReal64EntryFromValue

Usage

ViStatus Ivi_GetViReal64EntryFromValue(ViReal64 value, IviRangeTablePtr rangeTable, ViReal64* discreteOrMinValue, ViReal64* maxValue, ViReal64* coercedValue, ViInt32* tableIndex, ViString* commandString, ViInt32* commandValue);

This function finds the first range table entry that applies to the ViReal64 value you specify. If the function finds an entry, it returns the contents of the entry. If it does not find an entry, it returns an IVI_ERROR_INVALID_VALUE error.

If the range table type is IVI_VAL_DISCRETE, function searches for a match on the discreteOrMinValue field of each entry.

If the range table type is IVI_VAL_RANGED or IVI_VAL_COERCED, the function searches until the value you specify falls within the range between the discreteOrMinValue and maxValue fields of an entry. The value falls within the range if is greater than or equal to the discreteOrMinValue and less than or equal to the maxValue.

The function performs all ViReal64 comparisons using a comparison precision of 14 decimal digits.

The function returns the discreteOrMinValue, maxValue, and coercedValue fields as ViReal64 values.

Parameters

Name	Туре	Description
value	ViReal64	Specify the value that you want to find in the range table.
rangeTable	IviRangeTablePtr	Specify the address of the range table in which to search for the value.
discreteOrMinValue	ViReal64*	If the search succeeds, this parameter returns the value of the discreteOrMinValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
maxValue	ViReal64*	If the search succeeds, this parameter returns the value of the maxValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
coercedValue	ViReal64*	Specify the coerced value that you want to find in the range table.
tableIndex	Vilnt32*	If the search succeeds, this parameter returns the 0-based index of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.
commandString	ViString*	If the search succeeds, this

		parameter returns the pointer in the cmdString field of the entry.
		Do not change the contents of the string.
		You can pass VI_NULL for the parameter if you are not interested in this value.
Value	Vilnt32*	If the search succeeds, this parameter returns the value of the cmdValue field of the entry.
		You can pass VI_NULL for the parameter if you are not interested in this value.

commandValue

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_GetViReal64Type

Usage

ViStatus Ivi_GetViReal64Type(ViReal64 viReal64Value, ViInt32* viReal64Type);

Obtains the type of a ViReal64 value.

Parameters			
Name	Туре	Description	
viReal64Value	ViReal64	Specify the ViReal64 value for which you want to determine the type.	
viReal64Type	Vilnt32*	Returns the type of the ViReal64 value you specify.	
		(0) IVI_VAL_TYPE_NORMAL -normal value	
		(1) IVI_VAL_TYPE_NAN	-Not a Number (NaN)
		(2) IVI_VAL_TYPE_PINF	-positive infinity
		(3) IVI_VAL_TYPE_NINF	-negative infinity

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_InstrSpecificErrorQueueSize

Usage

ViStatus Ivi_InstrSpecificErrorQueueSize(ViSession vi, ViInt32* errorQueueSize);

This returns the number of entries currently in the instrument-specific error queue.

Use the instrument-specific error queue if querying the instrument for its status causes the instrument to lose the error value. In your check status callback, call Ivi_QueueInstrSpecificError to insert the instrument error code in the queue, and then return the IVI_ERROR_INSTR_SPECIFIC error code from the callback. In your PREFIX_error_query function, call Ivi_InstrSpecificErrorQueueSize to determine if there is an error in the queue. If not, invoke the check status callback directly. In either case, if there is an error, call Ivi_DequeueInstrSpecificError to retrieve the it.
Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
errorQueueSize	Vilnt32*	Returns the number of errors currently in the error queue.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_InterchangeCheck

Usage

ViBoolean = Ivi_InterchangeCheck(ViSession vi);

This function returns the current value of the IVI_ATTR_INTERCHANGE_CHECK attribute for the session you specify.

Use Ivi_InterchangeCheck in the high-level functions in class instrument drivers. Ivi_InterchangeCheck provides fast, convenient access to the IVI_ATTR_INTERCHANGE_CHECK attribute because it does no error checking and does not lock the session.



Note Do not call this function unless you have already locked the session.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the IVI_ATTR_INTERCHANGE_CHECK attribute. This attribute determines whether class drivers perform interchangeability checking. The specification for each instrument class defines the rules for interchangeability checking for that class.

Values:

VI_TRUE (1) Interchange checking on

VI_FALSE (0) Interchange checking off

If you pass an invalid session handle to the function, this parameter returns $\rm VI_FALSE.$

Related Topic

Ivi_InvalidateAllAttributes

Usage

ViStatus Ivi_InvalidateAllAttributes(ViSession vi);

This function invalidates the cache values of all instances of all attributes for the session.

Invalidating a cache value for an attribute ensures that the next call to an Ivi_GetAttribute or Ivi_SetAttribute function on the attribute invokes the read or write callback for the attribute.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_InvalidateAttribute

Usage

ViStatus Ivi_InvalidateAttribute(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID);

This function marks the cache value of an attribute as invalid. This ensures that the next call to an Ivi_GetAttribute or Ivi_SetAttribute function on the attribute invokes the read or write callback for the attribute.

For a repeated capability-based attribute, you can invalidate the attribute on a specific repeated capability or on all repeated capabilities.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_IOSession

Usage

ViSession = Ivi_IOSession(ViSession vi);

This function returns the current value of the IVI_ATTR_IO_SESSION attribute for the session you specify.

Use Ivi_IOSession in the high-level functions in specific instrument drivers. Ivi_IOSession provides fast, convenient access to the

IVI_ATTR_IO_SESSION attribute because it does no error checking and does not lock the session.



Note Do not call this function unless you have already locked the session.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the IVI_ATTR_IO_SESSION attribute for the session. You use the I/O session handle to communicate with the actual instrument.

If you pass an invalid session handle to the function, this parameter returns $\rm VI_NULL.$

Related Topic

Ivi_LockSession

Usage

ViStatus Ivi_LockSession(ViSession vi, ViBoolean* callerHasLock);

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.

You can use this function to protect a section of code which requires exclusive access to the instrument. This occurs when you take multiple actions that affect the instrument and you want to ensure that other execution threads do not disturb the instrument state until all of your actions execute. For example, if you set various instrument attributes and then trigger a measurement, you must guarantee that no other execution thread modifies the attribute values until you finish taking the measurement.

You can safely make nested calls to Ivi_LockSession within the same thread. To completely unlock the session, you must balance each call to Ivi_LockSession with a call to Ivi_UnlockSession. If, however, you use the Caller Has Lock parameter in all calls to Ivi_LockSession and Ivi_UnlockSession within a function, the IVI engine locks the session only once within the function regardless of the number of calls you make to Ivi_LockSession. This allows you to call Ivi_UnlockSession just once at the end of the function.

User applications, instrument drivers, and the IVI engine functions all have the ability to obtain a lock. The IVI engine functions always release the lock before they return. Instrument driver functions should do the same.

Instrument drivers export this function to the end-user through the PREFIX_LockSession function.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
callerHasLock	ViBoolean*	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. Initialize the local variable to VI_FALSE when you declare it. Pass the same address to any other calls you make to Ivi_LockSession or Ivi_UnlockSession in the same function.
		The parameter is an input/output parameter. Ivi_LockSession and Ivi_UnlockSession each inspect the current value and take the following actions:
		If the value is VI_TRUE, Ivi_LockSession does not lock the session again. If the value is VI_FALSE, Ivi_LockSession obtains the lock and sets the value of the parameter to VI_TRUE. If the value is VI_FALSE, Ivi_UnlockSession does not attempt to unlock the session. If the value is VI_TRUE, Ivi_UnlockSession unlocks the lock and sets the value of the parameter to VI_FALSE.

Thus, you can, call Ivi_UnlockSession at the

end of your function without worrying about whether you actually have the lock.

Example:

ViStatus PREFIX_Func (ViSession vi, ViInt32 flags){

```
ViStatus error = VI_SUCCESS;
ViBoolean haveLock = VI_FALSE;
```

if (flags & BIT_1)

```
{
viCheckErr( Ivi_LockSession(vi,
&haveLock));
viCheckErr( TakeAction1(vi));
```

if (flags & BIT_2)

```
{
viCheckErr( Ivi_UnlockSession(vi,
&haveLock));
viCheckErr( TakeAction2(vi));
viCheckErr( Ivi_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.

```
*/
```

Ivi_UnlockSession(vi, &haveLock);
return error;

}

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_NeedToCheckStatus

Usage

ViBoolean = Ivi_NeedToCheckStatus(ViSession vi);

This function returns an indication of whether the instrument driver has interacted with the instrument since the last time the IVI engine or the driver checked the status of the instrument.

Typically, the PREFIX_CheckStatus function that is internal to an instrument driver calls Ivi_NeedToCheckStatus to help determine whether it is necessary to invoke the check status callback for the session.

The IVI engine maintains an internal needToCheckStatus variable for each session indicating whether it is necessary to check the status of the instrument. When you create a new session, the initial value of the variable is VI_TRUE. The IVI engine sets the needToCheckStatus variable to VI_TRUE when it invokes the read callback or write callback for an attribute for which the IVI_VAL_DONT_CHECK_STATUS flag is 0. The Ivi_WriteInstrData and Ivi_WriteFromFile functions also set the variable to VI_TRUE. The IVI engine sets the variable to VI_TRUE. The IVI engine sets the variable to VI_TRUE with the IVI_VAL_DONT_CHECK_STATUS flag is 0. The Ivi_WriteInstrData and Ivi_WriteFromFile functions also set the variable to VI_TRUE. The IVI engine sets the variable to VI_FALSE after it invokes the check status callback successfully.

The Ivi_SetNeedToCheckStatus function allows an instrument driver to set the state of the internal needToCheckStatus variable. A driver typically sets the variable to VI_TRUE before it attempts direct instrument I/O. It sets it to VI_FALSE after it calls the check status callback successfully.

Ivi_NeedToCheckStatus returns the value of the internal needToCheckStatus variable. If the vi parameter is invalid, Ivi_NeedToCheckStatus returns VI_FALSE.



Note Do not call this function unless you have already locked the session.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the Ivi_NeedToCheckStatus variable for the session. Values:

VI_TRUE (1) Need to check status

VI_FALSE(0) No need to check status

If you pass an invalid session handle to the function, this parameter returns $\rm VI_FALSE.$

Related Topic

lvi_QueryInstrStatus

Usage

ViBoolean = Ivi_QueryInstrStatus(ViSession vi);

This function returns the current value of the

IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session you specify.

Use Ivi_QueryInstrStatus in the high-level functions in specific instrument drivers. Ivi_QueryInstrStatus provides fast, convenient access to the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute because it does no error checking and does not lock the session.



Note Do not call this function unless you have already locked the session.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session. This attribute determines whether or not to query the instrument error status after each operation.

Values:

VI_TRUE (1) Query instrument status

VI_FALSE (0) Do not query instrument status

If you pass an invalid session handle to the function, this parameter returns $\rm VI_FALSE.$

Related Topic

Ivi_QueueInstrSpecificError

Usage

ViStatus Ivi_QueueInstrSpecificError(ViSession vi, ViInt32 instrumentError, ViChar errorMessage[]);

This function inserts a new entry at the end of the instrument-specific error queue. The instrument-specific error queue is a software record of the error values you retrieve from the instrument.

Use the instrument-specific error queue if querying the instrument for its status causes the instrument to lose the error value. In your check status callback, call Ivi_QueueInstrSpecificError to insert the instrument error code in the queue, and then return the IVI_ERROR_INSTR_SPECIFIC error code from the callback. In your PREFIX_error_query function, call Ivi_InstrSpecificErrorQueueSize to determine if there is an error in the queue. If not, invoke the check status callback directly. In either case, if there is an error, call Ivi_DequeueInstrSpecificError to retrieve the it.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
instrumentError	Vilnt32	Pass the numeric error code to insert at the end of the instrument-specific error queue.
errorMessage	ViChar[]	Pass the error description string to insert at the end of the instrument-specific error queue.
Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_RangeChecking

Usage

ViBoolean = Ivi_RangeChecking(ViSession vi);

This function returns the current value of the IVI_ATTR_RANGE_CHECK attribute for the session you specify.

Use Ivi_RangeChecking in the high-level functions in specific instrument drivers. Ivi_RangeChecking provides fast, convenient access to the IVI_ATTR_RANGE_CHECK attribute because it does no error checking and does not lock the session.



Note Do not call this function unless you have already locked the session.

Parameters

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the IVI_ATTR_RANGE_CHECK attribute for the session. This attribute determines whether or not to range check parameters to instrument driver functions

Values:

VI_TRUE (1) Range check

VI_FALSE (0) Do not range check

If you pass an invalid session handle to the function, this parameter returns $\rm VI_FALSE.$

Related Topic

Ivi_RangeTableFree

Usage

ViStatus Ivi_RangeTableFree(ViSession vi, IviRangeTablePtr rangeTable, ViBoolean freeCommandStrings);

This function deallocates a range table you create dynamically with Ivi_RangeTableNew. It calls Ivi_Free to free the IviRangeTable structure and the array of IviRangeTableEntry structures. It optionally frees the cmdString field in each entry.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
rangeTable	IviRangeTablePtr	Specify the table pointer you obtain from Ivi_RangeTableNew.
freeCommandStrings	ViBoolean	Specify whether you want the function to deallocate the command strings in the range table. If you pass VI_TRUE (1), the function calls Ivi_Free on the cmdString field of each entry. Do not pass VI_TRUE unless you allocated the command strings using Ivi_Alloc. Pass VI_FALSE (0) if you do not want the function to deallocate the command strings.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_RangeTableNew

Usage

ViStatus Ivi_RangeTableNew(ViSession vi, ViInt32 numberOfEntries, ViInt32 typeOfTable, ViBoolean hasMinimum, ViBoolean hasMaximum, IviRangeTablePtr* rangeTable);

This function dynamically allocates a range table. Range tables you create with this function are called "dynamic range tables". Range tables you define statically in your source code are called "static range tables".

If the values in the range table for a particular attribute can change depending on the settings of other attributes, you must create it as a dynamic range table. You must also install a range table callback using Ivi_SetAttrRangeTableCallback. In the range table callback, you modify the contents of the range table and then return its address.

To allow for multithreading and multiple sessions to the same instrument type, you must create a separate dynamic range table for each IVI session. It is convenient to pass the address of the dynamic range table to Ivi_AddAttributeViInt32 or Ivi_AddAttributeViReal64 when you create the attribute. Your range table callback can then use the Ivi_GetStoredRangeTablePtr function to obtain the address of the dynamic range table for the session.

This function allocates the IviRangeTable structure and an array of IviRangeTableEntry structures. It allocates space in the array for the number of entries you specify, which must include the termination entry. It sets the last entry as the termination entry. Use the Ivi SetRangeTableEntry function to set the values within the entries.

If the number of entries in the table varies, specify the maximum number of entries that it can contain. Use the Ivi_SetRangeTableEnd function to change the location of the termination entry.

The IVI engine keeps track of the memory you allocate with this function in each session. It automatically frees the memory when you call Ivi_Dispose on the session.

If you want to deallocate the table before the session ends, call the Ivi_RangeTableFree function.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
numberOfEntries	Vilnt32	Specify the number of entries you want in the range table, including the termination entry.
		For example, if you want to have 6 entries in the range table, excluding the termination entry, pass 7 for this parameter.
		The function automatically sets the last entry as the termination entry. To set an earlier entry as the termination entry, call Ivi_SetRangeTableEnd.
typeOfTable	Vilnt32	Specify the type of range table you want to create. The type indicates how the IVI engine interprets the discreteOrMinValue, maxValue, and coercedValue fields in each entry.
		Valid Values:
		(0) IVI_VAL_DISCRETE
		(1) IVI_VAL_RANGED
		(2) IVI_VAL_COERCED
		(1) Discrete—Each table entry defines a discrete value. The discreteOrMinValue field contains

the discrete value. The maxValue and coercedValue fields are not used.

(2) Ranged—Each table entry defines a range with a minimum and a maximum value. The discreteOrMinValue field holds the minimum value, and the maxValue field holds the maximum value. The coercedValue field is not used. If the attribute has only one continuous valid range and you do not assign different command strings or command values to subsets of the range, create the range table with only one entry other than the terminating entry.

(3) Coerced—Each table entry defines a discrete value that represents a range of values. This is useful when an instrument supports a set of ranges, each of which you must specify to the instrument with one discrete value. The discreteOrMinValue holds the minimum value of the range, maxValue holds the maximum value, and coercedValue holds the discrete value that represents the range.

hasMinimum

ViBoolean

Indicates whether the table contains a meaningful minimum value. Pass VI_TRUE (1) if the range table has a meaningful minimum valid value. Otherwise, pass VI_FALSE (0).

		Note For tables with type IVI_VAL_COERCED, the minimum value represents the minimum coerced value.
hasMaximum	ViBoolean	Indicates whether the table contains a meaningful maximum value. Pass VI_TRUE (1) if the range table has a meaningful maximum valid value. Otherwise, pass VI_FALSE (0).
		Note For tables with type IVI_VAL_COERCED, the maximum value represents the maximum coerced value.
rangeTable	IviRangeTablePtr*	[*] Returns a pointer to the range table the function dynamically allocates. Use Ivi_SetRangeTableEntry to configure each entry after calling this function.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_ReadInstrData

Usage

ViStatus Ivi_ReadInstrData(ViSession vi, ViInt32 numberBytesToRead, ViChar readBuffer[], ViInt32* numBytesRead);

This function reads data directly from an instrument using VISA I/O. The function bypasses the attribute state caching mechanism. Use this function only to implement the PREFIX_ReadInstrData function that your instrument driver exports to the end-user.

The function assumes that the IVI_ATTR_IO_SESSION attribute for the IVI session you specify holds a valid VISA session for the instrument.

If data is not available at the instrument's output buffer when you call this function, the instrument might hang up. In that case, the function does not return until the VISA I/O call times out. If you have disabled the timeout, the function hangs indefinitely.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
numberBytesToRead	Vilnt32	Specify the maximum number of bytes to read from the instrument. The Read Buffer parameter must be a ViChar buffer that contains at least the number of bytes you specify.
		If the number of bytes you specify is less than the number of bytes in the instrument's output buffer, you must call this function again to empty the output buffer. If you do not empty the instrument's output buffer, the instrument might return invalid data in response to subsequent requests.
		If data is not available at the instrument's output buffer when you call this function, the instrument might hang up. In that case, the function does not return until the VISA I/O call times out. If you have disabled the timeout, the function hangs indefinitely.
readBuffer	ViChar[]	A buffer in which the function places the data it receives from the instrument.
		The buffer must be a ViChar array that has at least as many bytes as you specify in the Number Bytes To Read parameter.

		This function does not write an ASCII NUL byte to terminate the data, nor does it clear the buffer beyond the bytes it actually receives from the instrument.
numBytesRead	Vilnt32*	This control returns the actual number of bytes the function received from the instrument. This is the value that the VISA viRead function returns.
		If the actual number of bytes received is less than the number of bytes you specify in the Number Bytes To Read parameter, the instrument's output buffer has probably emptied.
		If the number of bytes received is 0, the most probable cause is that no data was available at the instrument's output buffer.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_ReadToFile

Usage

ViStatus Ivi_ReadToFile(ViSession vi, ViChar filename[], ViInt32 readNumber_ofBytes, ViInt32 fileAction, ViInt32* returnCount);

This function reads data from an instrument using VISA I/O and writes it to a file you specify. Use this function internally in your instrument driver.

The function assumes that the IVI_ATTR_IO_SESSION attribute for the IVI session you specify holds a valid VISA session for the instrument.

The function opens the file in binary mode.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
filename	ViChar[]	Specify the pathname of the file from which to write the data. You can specify an absolute pathname, a relative pathname, or a simple filename. The function treats relative pathnames and simple filenames as relative to the current working directory.
		If you enter a literal string in this parameter under Windows, be sure to use double backslashes to represent one backslash in the pathname.
readNumber_ofBytes	Vilnt32	Specify the maximum number of bytes to read from the instrument.
fileAction	Vilnt32	Specify whether you want the function to append the data it receives from the instrument to an existing file or to create a new file. Values:
		(1) IVI_VAL_TRUNCATE
		(2) IVI_VAL_APPEND
		(1) Truncate—If the file already exists, delete its contents and write the instrument data to it. If the file does not exist, create it and write the

instrument data to it.

(2) Append—If the file already exists, append the instrument data to it. If the file does not exist, create it and write the instrument data to it.

returnCount

Vilnt32*

Returns the number of bytes the function successfully writes from the file.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_ResetInterchangeCheck

Usage

ViStatus Ivi_ResetInterchangeCheck(ViSession vi);

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 Ivi_GetNextInterchangeCheckString 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 Ivi_GetNextInterchangeCheckString function until no more interchangeability warnings are returned. If you are not interested in the content of those warnings, you can call the Ivi_ClearInterchangeWarnings function.

Parameters

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_RestrictAttrToChannels

Usage

ViStatus Ivi_RestrictAttrToChannels(ViSession vi, ViAttr attributeID, ViChar ChannelStringsList[]);

This function restricts an attribute to specific channels, thereby excluding you from using the attribute on other channels. You can call this function only on attributes for which you have enabled the IVI_VAL_MULTI_CHANNEL flag.

When you initially add an attribute, it applies to all channels. If you want it to apply to only a subset, call this function. This function can only be called once after an attribute has been created.

Example:

Ivi_BuildChannelTable (vi, "1,2,3,4", VI_FALSE, VI_NULL);
Ivi_RestrictAttrToChannels (vi, PREFIX_ATTR_RANGE, "1,2");

As a result of these function calls, PREFIX_ATTR_RANGE is valid only for channels "1" and "2".

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces

the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

ChannelStringsList ViChar[] Pass a list of the channel strings to which you want to restrict the attribute you specify. You must separate channel strings with commas. You can include spaces after the commas.

For example, if you call Ivi_BuildChannelTable with the channel string "1,2,3,4", then pass "1,2" for this parameter to restrict the attribute you specify to channels "1" and "2".

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_RestrictAttrToInstances

Usage

ViStatus Ivi_RestrictAttrToInstances(ViSession vi, ViAttr attributeID, ViChar instances[]);
Purpose

This function restricts an attribute to specific repeated capability instances, thereby excluding you from using the attribute on other repeated capability instances.

You can call this function only on attributes that are defined for a repeated capability. When you initially add an attribute, it applies to all repeated capability instances. If you want it to apply to only a subset, call this function.

This function can only be called once after an attribute has been created. Example:

Ivi_BuildRepCapTable (vi, "Marker", "1,2,3,4");

Ivi_RestrictAttrToInstances (vi, PREFIX_ATTR_RANGE, "1,2");

As a result of these function calls, PREFIX_ATTR_RANGE is valid only for markers "1" and "2".

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user- accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:
		#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define FL45_ATTR_HOLD_THRESHOLD \
(IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

instances ViChar[] Pass a list of the repeated capability identifiers to which you want to restrict the attribute you specify. You must separate identifiers with commas. You can include spaces after the commas.

For example, if you call Ivi_BuildRepCapTable with the string "1,2,3,4", passing "1,2" for this parameter restricts the attribute to the repeated capability instances "1" and "2".

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCheckCallbackViAddr

Usage

ViStatus Ivi_SetAttrCheckCallbackViAddr(ViSession vi, ViAttr attributeID, CheckAttrViAddr_CallbackPtr checkCallback);

Purpose

This function sets the check callback function for a ViAddr attribute. The IVI engine calls the check callback function to validate new values to which you attempt to set the attribute.

If you do not want the IVI engine to invoke a check callback for the attribute, pass VI_NULL for the Check Callback parameter.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that from Ivi_SpecificDriverNew identifies a particular IVI s
attributeID	ViAttr	Pass the ID of an attribute parameter.
		The include file for a speci driver defines constant na the user-accessible attribu- to the driver. This includes that the IVI engine defines that the instrument class c attributes that are specific particular instrument. Eacl constant name begins with PREFIX_ATTR_, where PF specific instrument prefix.
		For each IVI engine attribut specific driver include file same constant name that ivi.h, except that the speci prefix replaces the IVI pre- example, ivi.h defines IVI_ATTR_CACHE, and th include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA(IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include fi the specific instrument pre

the class prefix. For exam class include file, ividmm.h IVIDMM_ATTR_RANGE, defines the following cons

#define FL45_ATTR_RAN IVIDMM_ATTR_RANGE

For each specific instrume the specific driver include constant name and assigr is an offset from IVI_SPECIFIC_PUBLIC_A

For example, fl45.h define constant name:

#define

FL45_ATTR_HOLD_THF (IVI_SPECIFIC_PUBLIC_ + 3L)

For each attribute that is p instrument driver, the instr source file defines a const assigns a value that is an IVI_SPECIFIC_PRIVATE_/ For example, hp34401a.c d following constant name:

#define

HP34401A_ATTR_TRIG((IVI_SPECIFIC_PRIVATI + 1L)

checkCallback CheckAttrViAddr_CallbackPtr Specify the check callback want the IVI engine to invo attribute values.

The function must have th prototype:

ViStatus _VI_FUNC Callbac vi,ViConstString repCapNan

ViAttr attributeId, ViAddr value);

Note If you want to IVI Specific Driver A dialog box to develo instrument driver so retain the parameter shown in the prototy callback.

If you do not want to use a callback function, pass VI

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCheckCallbackViBoolean

Usage

ViStatus Ivi_SetAttrCheckCallbackViBoolean(ViSession vi, ViAttr attributeID, CheckAttrViBoolean_CallbackPtr checkCallback);

Purpose

This function sets the check callback function for a ViBoolean attribute. The IVI engine calls the check callback function to validate new values to which you attempt to set the attribute.

If you do not want the IVI engine to invoke a check callback for the attribute, pass VI_NULL for the Check Callback parameter.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle from Ivi_SpecificDriver identifies a particular IV
attributeID	ViAttr	Pass the ID of an attrik parameter.
		The include file for a s driver defines constant the user-accessible att to the driver. This inclu that the IVI engine defi that the instrument cla attributes that are spec particular instrument. E constant name begins PREFIX_ATTR_, where specific instrument pre
		For each IVI engine at specific driver include same constant name t ivi.h, except that the sp prefix replaces the IVI example, ivi.h defines IVI_ATTR_CACHE, an include file, fl45.h, defines
		#define FL45_ATTR_(IVI_ATTR_CACHE
		For each instrument cl specific driver include same constant name t instrument class incluc the specific instrument

the class prefix. For ex class include file, ividn IVIDMM_ATTR_RANC defines the following c

#define FL45_ATTR_I IVIDMM_ATTR_RAN

For each specific instruction the specific driver inclucion constant name and assist an offset from IVI_SPECIFIC_PUBLIC For example, fl45.h deconstant name:

#define FL45_ATTR_HOLD_' (IVI_SPECIFIC_PUB) + 3L)

For each attribute that instrument driver, the i source file defines a co assigns a value that is IVI_SPECIFIC_PRIVAT For example, hp34401a following constant nar

#define
HP34401A_ATTR_TR
(IVI_SPECIFIC_PRIV
+ 1L)

checkCallback CheckAttrViBoolean_CallbackPtr Specify the check callt want the IVI engine to attribute values.

The function must hav prototype:

ViStatus _VI_FUNC Cal vi,ViConstString repCap

ViAttr attributeId, ViBoolean value);

If you want to use the I Driver Attributes dialoc your instrument driver retain the parameter na the prototype for the ca

If you do not want to us callback function, pass

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCheckCallbackViInt32

Usage

ViStatus Ivi_SetAttrCheckCallbackViInt32(ViSession vi, ViAttr attributeID, CheckAttrViInt32_CallbackPtr checkCallback);

Purpose

This function sets the check callback function for a ViInt32 attribute. The IVI engine calls the check callback function to validate new values to which you attempt to set the attribute.

The IVI engine automatically installs its default check callback when you create the attribute.

If you do not want the IVI engine to invoke a check callback for the attribute, pass VI_NULL for the Check Callback parameter.

If you want to specify your own callback function but you want to use the default check callback within your function, you can call Ivi_DefaultCheckCallbackViInt32.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tha from Ivi_SpecificDriverNev identifies a particular IVI s
attributeID	ViAttr	Pass the ID of an attribute parameter.
		The include file for a spec driver defines constant na the user-accessible attribu- to the driver. This includes that the IVI engine defines that the instrument class of attributes that are specific particular instrument. Eac constant name begins with PREFIX_ATTR_, where PI specific instrument prefix.
		For each IVI engine attribus specific driver include file same constant name that ivi.h, except that the spec prefix replaces the IVI pre example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA(IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include fi the specific instrument pre

the class prefix. For exam class include file, ividmm.l IVIDMM_ATTR_RANGE, defines the following cons

#define FL45_ATTR_RAI IVIDMM_ATTR_RANGE

For each specific instrume the specific driver include constant name and assign is an offset from

IVI_SPECIFIC_PUBLIC_A For example, fl45.h define constant name:

#define

FL45_ATTR_HOLD_THI (IVI_SPECIFIC_PUBLIC + 3L)

For each attribute that is p instrument driver, the instrusource file defines a const assigns a value that is an IVI_SPECIFIC_PRIVATE_/ For example, hp34401a.c c following constant name:

#define
HP34401A_ATTR_TRIG(
(IVI_SPECIFIC_PRIVATI
+ 1L)

checkCallback CheckAttrViInt32_CallbackPtr Specify the check callback want the IVI engine to inverse attribute values.

The function must have th prototype:

ViStatus _VI_FUNC Callba vi,ViConstString repCapNar

ViAttr attributeId, ViInt32 value);

Note If you want to IVI Specific Driver A dialog box to develc instrument driver so retain the paramete shown in the prototy callback.

If you do not want to use a callback function, pass VI

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCheckCallbackViInt64

Usage

ViStatus Ivi_SetAttrCheckCallbackViInt64(ViSession vi, ViAttr attributeID, CheckAttrViInt64_CallbackPtr checkCallback);

Purpose

This function sets the check callback function for a ViInt64 attribute. The IVI engine calls the check callback function to validate new values to which you attempt to set the attribute.

The IVI engine automatically installs its default check callback when you create the attribute.

If you do not want the IVI engine to invoke a check callback for the attribute, pass VI_NULL for the Check Callback parameter.

If you want to specify your own callback function but you want to use the default check callback within your function, you can call Ivi_DefaultCheckCallbackViInt64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tha from Ivi_SpecificDriverNev identifies a particular IVI s
attributeID	ViAttr	Pass the ID of an attribute parameter.
		The include file for a spec driver defines constant na the user-accessible attribu- to the driver. This includes that the IVI engine defines that the instrument class of attributes that are specific particular instrument. Eac constant name begins with PREFIX_ATTR_, where PI specific instrument prefix.
		For each IVI engine attribus specific driver include file same constant name that ivi.h, except that the spec prefix replaces the IVI pre example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA(IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include fi the specific instrument pre

the class prefix. For exam class include file, ividmm.l IVIDMM_ATTR_RANGE, defines the following cons

#define FL45_ATTR_RAI IVIDMM_ATTR_RANGE

For each specific instrume the specific driver include constant name and assign is an offset from

IVI_SPECIFIC_PUBLIC_A For example, fl45.h define constant name:

#define

FL45_ATTR_HOLD_THI (IVI_SPECIFIC_PUBLIC + 3L)

For each attribute that is p instrument driver, the instrusource file defines a const assigns a value that is an IVI_SPECIFIC_PRIVATE_1 For example, hp34401a.c c following constant name:

#define
HP34401A_ATTR_TRIG(
(IVI_SPECIFIC_PRIVATI
+ 1L)

checkCallback CheckAttrViInt64_CallbackPtr Specify the check callback want the IVI engine to inverse attribute values.

The function must have th prototype:

ViStatus _VI_FUNC Callba vi,ViConstString repCapNar

ViAttr attributeId, ViInt64 value);

Note If you want to IVI Specific Driver A dialog box to develc instrument driver so retain the paramete shown in the prototy callback.

If you do not want to use a callback function, pass VI

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCheckCallbackViReal64

Usage

ViStatus Ivi_SetAttrCheckCallbackViReal64(ViSession vi, ViAttr attributeID, CheckAttrViReal64_CallbackPtr checkCallback);

Purpose

This function sets the check callback function for a ViReal64 attribute. The IVI engine calls the check callback function to validate new values to which you attempt to set the attribute.

The IVI engine automatically installs its default check callback when you create the attribute.

If you do not want the IVI engine to invoke a check callback for the attribute, pass VI_NULL for the Check Callback parameter.

If you want to specify your own callback function but you want to use the default check callback within your function, you can call Ivi_DefaultCheckCallbackViReal64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tl from Ivi_SpecificDriverN identifies a particular IV
attributeID	ViAttr	Pass the ID of an attribu parameter.
		The include file for a sp driver defines constant the user-accessible attri- to the driver. This includ that the IVI engine defin that the instrument class attributes that are speci particular instrument. Ea constant name begins v PREFIX_ATTR_, where specific instrument prefi
		For each IVI engine attr specific driver include fil same constant name th ivi.h, except that the spe prefix replaces the IVI p example, ivi.h defines IVI_ATTR_CACHE, and include file, fl45.h, defin constant name:
		#define FL45_ATTR_C IVI_ATTR_CACHE
		For each instrument cla specific driver include fil same constant name th instrument class include the specific instrument p

the class prefix. For exa class include file, ividmi IVIDMM_ATTR_RANGI defines the following co

#define FL45_ATTR_R IVIDMM_ATTR_RAN

For each specific instrue the specific driver incluc constant name and assi is an offset from IVI_SPECIFIC_PUBLIC_ For example, fl45.h defi constant name:

#define FL45_ATTR_HOLD_T (IVI_SPECIFIC_PUBL + 3L)

For each attribute that is instrument driver, the in source file defines a cor assigns a value that is a IVI_SPECIFIC_PRIVATE For example, hp34401a.(following constant name

#define
HP34401A_ATTR_TRI
(IVI_SPECIFIC_PRIVA
+ 1L)

checkCallback CheckAttrViReal64_CallbackPtr Specify the check callback want the IVI engine to ir attribute values. The function must have

prototype:

ViStatus _VI_FUNC Calll vi,ViConstString repCapN

ViAttr attributeId, ViReal64 value);

Note If you want IVI Specific Driver dialog box to deve instrument driver retain the parame shown in the prote callback.

If you do not want to use callback function, pass

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCheckCallbackViSession

Usage

ViStatus Ivi_SetAttrCheckCallbackViSession(ViSession vi, ViAttr attributeID, CheckAttrViSession_CallbackPtr checkCallback);

Purpose

This function sets the check callback function for a ViSession attribute. The IVI engine calls the check callback function to validate new values to which you attempt to set the attribute.

If you do not want the IVI engine to invoke a check callback for the attribute, pass VI_NULL for the Check Callback parameter.
Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle from Ivi_SpecificDriverI identifies a particular IV
attributeID	ViAttr	Pass the ID of an attrik parameter.
		The include file for a sp driver defines constant the user-accessible att to the driver. This inclu that the IVI engine defi that the instrument clas attributes that are spec particular instrument. E constant name begins PREFIX_ATTR_, where specific instrument pre
		For each IVI engine att specific driver include 1 same constant name tl ivi.h, except that the sp prefix replaces the IVI example, ivi.h defines IVI_ATTR_CACHE, an include file, fl45.h, defin constant name:
		#define FL45_ATTR_(IVI_ATTR_CACHE
		For each instrument cla specific driver include 1 same constant name tl instrument class includ the specific instrument

the class prefix. For ex class include file, ividm IVIDMM_ATTR_RANC defines the following co

#define FL45_ATTR_I IVIDMM_ATTR_RAN

For each specific instruction the specific driver inclucion constant name and assess is an offset from IVI_SPECIFIC_PUBLIC For example, fl45.h det constant name:

#define
FL45_ATTR_HOLD_7
(IVI_SPECIFIC_PUBI
+ 3L)

For each attribute that instrument driver, the in source file defines a cc assigns a value that is IVI_SPECIFIC_PRIVAT For example, hp34401a following constant nam

#define HP34401A_ATTR_TR (IVI_SPECIFIC_PRIV + 1L)

checkCallback CheckAttrViSession_CallbackPtr Specify the check call want the IVI engine to attribute values.

The function must have prototype:

ViStatus _VI_FUNC Cal vi,ViConstString repCap

ViAttr attributeId, ViSession value);

Note If you wan IVI Specific Drive dialog box to dev instrument driver retain the param shown in the pro callback.

If you do not want to us callback function, pass

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCheckCallbackViString

Usage

ViStatus Ivi_SetAttrCheckCallbackViString(ViSession vi, ViAttr attributeID, CheckAttrViString_CallbackPtr checkCallback);

Purpose

This function sets the check callback function for a ViString attribute. The IVI engine calls the check callback function to validate new values to which you attempt to set the attribute.

If you do not want the IVI engine to invoke a check callback for the attribute, pass VI_NULL for the Check Callback parameter.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle the from Ivi_SpecificDriverNe identifies a particular IVI
attributeID	ViAttr	Pass the ID of an attribut parameter.
		The include file for a spendriver defines constant network the user-accessible attributes to the driver. This include that the IVI engine define that the instrument class attributes that are specific particular instrument. Eac constant name begins wi PREFIX_ATTR_, where P specific instrument prefix
		For each IVI engine attrik specific driver include file same constant name tha ivi.h, except that the spec prefix replaces the IVI pro example, ivi.h defines IVI_ATTR_CACHE, and t include file, fl45.h, define constant name:
		#define FL45_ATTR_CA IVI_ATTR_CACHE
		For each instrument clas specific driver include file same constant name tha instrument class include the specific instrument pr

the class prefix. For exar class include file, ividmm IVIDMM_ATTR_RANGE, defines the following con

#define FL45_ATTR_RA IVIDMM_ATTR_RANG

For each specific instrum the specific driver include constant name and assig is an offset from IVI_SPECIFIC_PUBLIC_*I* For example, fl45.h defin constant name:

#define

FL45_ATTR_HOLD_TH (IVI_SPECIFIC_PUBLI(+ 3L)

For each attribute that is instrument driver, the ins source file defines a cons assigns a value that is ar IVI_SPECIFIC_PRIVATE_ For example, hp34401a.c following constant name:

#define HP34401A_ATTR_TRIG (IVI_SPECIFIC_PRIVAT + 1L)

checkCallback CheckAttrViString_CallbackPtr Specify the check callbac want the IVI engine to inv attribute values.

The function must have t prototype:

ViStatus _VI_FUNC Callbavi,ViConstString repCapNa

ViAttr attributeId, ViConstString value);

Note If you want to IVI Specific Driver, dialog box to devel instrument driver so retain the paramete shown in the protot callback.

If you do not want to use callback function, pass $\ensuremath{\mathrm{V}}$

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCoerceCallbackViAddr

Usage

ViStatus Ivi_SetAttrCoerceCallbackViAddr(ViSession vi, ViAttr attributeID, CoerceAttrViAddr_CallbackPtr coerceCallback);

Purpose

This function sets the coerce callback function for a ViAddr attribute. The IVI engine calls the coerce callback function when you attempt to set the attribute to a new value. The job of the coerce callback is to convert the value you specify into the value to send to the instrument. The IVI engine invokes the coerce callback after it invokes the check callback.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle th from Ivi_SpecificDriverNe identifies a particular IVI
attributeID	ViAttr	Pass the ID of an attribu parameter.
		The include file for a spe driver defines constant r the user-accessible attril to the driver. This include that the IVI engine define that the instrument class attributes that are specif particular instrument. Ea constant name begins w PREFIX_ATTR_, where I specific instrument prefix
		For each IVI engine attri specific driver include file same constant name tha ivi.h, except that the spe prefix replaces the IVI p example, ivi.h defines IVI_ATTR_CACHE, and include file, fl45.h, define constant name:
		#define FL45_ATTR_C# IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include the specific instrument p

the class prefix. For exa class include file, ividmn IVIDMM_ATTR_RANGE defines the following cor

#define FL45_ATTR_R/ IVIDMM_ATTR_RAN(

For each specific instrum the specific driver includ constant name and assign is an offset from IVI_SPECIFIC_PUBLIC_ For example, fl45.h defir constant name:

#define
FL45_ATTR_HOLD_TF
(IVI_SPECIFIC_PUBLI
+ 3L)

For each attribute that is instrument driver, the ins source file defines a con assigns a value that is a IVI_SPECIFIC_PRIVATE For example, hp34401a.c following constant name

#define

HP34401A_ATTR_TRI((IVI_SPECIFIC_PRIVA' + 1L)

coerceCallback CoerceAttrViAddr_CallbackPtr Specify the coerce callback want the IVI engine to in attempt to set the attribu value.

prototype:

ViStatus _VI_FUNC Callb

vi,ViConstString repCapN, ViAttr attributeId, ViAddr value, ViAddr *coercedValue);

Note If you want 1 IVI Specific Driver dialog box to deve instrument driver s retain the paramet shown in the proto callback.

If you do not want to use callback function, pass \mathbf{N}

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCoerceCallbackViBoolean

Usage

ViStatus Ivi_SetAttrCoerceCallbackViBoolean(ViSession vi, ViAttr attributeID, CoerceAttrViBoolean_CallbackPtr coerceCallback);

Purpose

This function sets the coerce callback function for a ViBoolean attribute. The IVI engine calls the coerce callback function when you attempt to set the attribute to a new value. The job of the coerce callback for a ViBoolean attribute is to convert the value you specify into either VI_TRUE (1) or VI_FALSE (0). The IVI engine invokes the coerce callback after it invokes the check callback.

The IVI engine automatically installs its default coerce callback when you create the attribute. The default callback coerces all non-zero values to VI_TRUE (1). If you do want the IVI engine to invoke a coerce callback for the attribute, pass VI_NULL for the Coerce Callback parameter.

If you want to specify your own callback function but you want to use the default coerce callback within your function, you can call Ivi_DefaultCoerceCallbackViBoolean.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handl from Ivi_SpecificDriv identifies a particular
attributeID	ViAttr	identifies a particular Pass the ID of an att parameter. The include file for a driver defines consta the user-accessible a to the driver. This inc that the IVI engine de that the instrument c attributes that are sp particular instrument constant name begir PREFIX_ATTR_, whe specific instrument p For each IVI engine a specific driver include same constant name ivi.h, except that the prefix replaces the IN example, ivi.h define IVI_ATTR_CACHE, a include file, fl45.h, de constant name: #define FL45_ATTR IVI_ATTR_CACHE For each instrument specific driver include
		instrument class inclution the specific instrume

the class prefix. For class include file, ivid IVIDMM ATTR RAP defines the following #define FL45 ATTR IVIDMM_ATTR_R/ For each specific ins the specific driver inc constant name and a is an offset from IVI_SPECIFIC_PUBL For example, fl45.h c constant name: #define FL45 ATTR HOLD (IVI SPECIFIC PU + 3L) For each attribute the instrument driver, the source file defines a assigns a value that IVI SPECIFIC PRIVA For example, hp3440 following constant na #define HP34401A ATTR 7 (IVI_SPECIFIC_PR + 1L) coerceCallback CoerceAttrViBoolean CallbackPtr Specify the coerce ca want the IVI engine t attempt to set the att value. The function must ha prototype: ViStatus _VI_FUNC C

ViConstString repCapl attributeId, ViBoolean *coercedValue);

If you want to use the Driver Attributes diale your instrument drive retain the parameter the prototype for the

If you do not want to callback function, pa:

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCoerceCallbackViInt32

Usage

ViStatus Ivi_SetAttrCoerceCallbackViInt32(ViSession vi, ViAttr attributeID, CoerceAttrViInt32_CallbackPtr coerceCallback);

Purpose

This function sets the coerce callback function for a VIInt32 attribute. The IVI engine calls the coerce callback function when you attempt to set the attribute to a new value. The job of the coerce callback is to convert the value you specify into the value to send to the instrument. The IVI engine invokes the coerce callback after it invokes the check callback.

The IVI engine automatically installs its default coerce callback when you create the attribute.

If you do not want the IVI engine to invoke a coerce callback for the attribute, pass VI_NULL for the Coerce Callback parameter.

If you want to specify your own callback function but you want to use the default coerce callback within your function, you can call Ivi_DefaultCoerceCallbackViInt32.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle the from Ivi_SpecificDriverNote identifies a particular IVI
attributeID	ViAttr	Pass the ID of an attribu parameter.
		The include file for a spe driver defines constant r the user-accessible attri to the driver. This includ that the IVI engine defin that the instrument class attributes that are specif particular instrument. Ea constant name begins w PREFIX_ATTR_, where specific instrument prefi
		For each IVI engine attri specific driver include fil same constant name tha ivi.h, except that the spe prefix replaces the IVI p example, ivi.h defines IVI_ATTR_CACHE, and include file, fl45.h, define constant name:
		#define FL45_ATTR_C/ IVI_ATTR_CACHE
		For each instrument class specific driver include fil same constant name that instrument class include the specific instrument p

the class prefix. For exa class include file, ividmn IVIDMM_ATTR_RANGI defines the following coi

#define FL45_ATTR_R/ IVIDMM_ATTR_RAN(

For each specific instrur the specific driver includ constant name and assi is an offset from IVI_SPECIFIC_PUBLIC_ For example, fl45.h defir constant name:

#define FL45_ATTR_HOLD_TI (IVI_SPECIFIC_PUBLI + 3L)

For each attribute that is instrument driver, the ins source file defines a cor assigns a value that is a IVI_SPECIFIC_PRIVATE For example, hp34401a.c following constant name

#define

HP34401A_ATTR_TRI((IVI_SPECIFIC_PRIVA + 1L)

coerceCallback CoerceAttrViInt32_CallbackPtr Specify the coerce callb want the IVI engine to ir attempt to set the attribu value.

The function must have prototype:

ViStatus _VI_FUNC Callt

vi,ViConstString repCapN ViAttr attributeId, ViInt32 value, ViInt32 *coercedValue);

Note If you want IVI Specific Driver dialog box to deve instrument driver s retain the paramel shown in the proto callback.

If you do not want to use callback function, pass ¹

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCoerceCallbackViInt64

Usage

ViStatus Ivi_SetAttrCoerceCallbackViInt64(ViSession vi, ViAttr attributeID, CoerceAttrViInt64_CallbackPtr coerceCallback);

Purpose

This function sets the coerce callback function for a VIInt64 attribute. The IVI engine calls the coerce callback function when you attempt to set the attribute to a new value. The job of the coerce callback is to convert the value you specify into the value to send to the instrument. The IVI engine invokes the coerce callback after it invokes the check callback.

The IVI engine automatically installs its default coerce callback when you create the attribute.

If you do not want the IVI engine to invoke a coerce callback for the attribute, pass VI_NULL for the Coerce Callback parameter.

If you want to specify your own callback function but you want to use the default coerce callback within your function, you can call Ivi_DefaultCoerceCallbackViInt64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle the from Ivi_SpecificDriverNote identifies a particular IVI
attributeID	ViAttr	Pass the ID of an attribu parameter.
		The include file for a spe driver defines constant r the user-accessible attri to the driver. This includ that the IVI engine defin that the instrument class attributes that are specif particular instrument. Ea constant name begins w PREFIX_ATTR_, where specific instrument prefi
		For each IVI engine attri specific driver include fil same constant name tha ivi.h, except that the spe prefix replaces the IVI p example, ivi.h defines IVI_ATTR_CACHE, and include file, fl45.h, define constant name:
		#define FL45_ATTR_C/ IVI_ATTR_CACHE
		For each instrument class specific driver include fil same constant name that instrument class include the specific instrument p

the class prefix. For exa class include file, ividmn IVIDMM_ATTR_RANGI defines the following coi

#define FL45_ATTR_R/ IVIDMM_ATTR_RAN(

For each specific instrur the specific driver includ constant name and assi is an offset from IVI_SPECIFIC_PUBLIC_ For example, fl45.h defir constant name:

#define FL45_ATTR_HOLD_TI (IVI_SPECIFIC_PUBLI + 3L)

For each attribute that is instrument driver, the ins source file defines a cor assigns a value that is a IVI_SPECIFIC_PRIVATE For example, hp34401a.c following constant name

#define

HP34401A_ATTR_TRI((IVI_SPECIFIC_PRIVA + 1L)

coerceCallback CoerceAttrViInt64_CallbackPtr Specify the coerce callb want the IVI engine to ir attempt to set the attribu value.

The function must have prototype:

ViStatus _VI_FUNC Callt

vi,ViConstString repCapN ViAttr attributeId, ViInt32 value, ViInt32 *coercedValue);

Note If you want IVI Specific Driver dialog box to deve instrument driver s retain the paramel shown in the proto callback.

If you do not want to use callback function, pass ¹

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCoerceCallbackViReal64

Usage

ViStatus Ivi_SetAttrCoerceCallbackViReal64(ViSession vi, ViAttr attributeID, CoerceAttrViReal64_CallbackPtr coerceCallback);

Purpose

This function sets the coerce callback function for a ViReal64 attribute. The IVI engine calls the coerce callback function when the you attempt to set the attribute to a new value. The job of the coerce callback is to convert the value you specify into the value to send to the instrument. The IVI engine invokes the coerce callback after it invokes the check callback.

The IVI engine automatically installs its default coerce callback when you create the attribute.

If you do not want the IVI engine to invoke a coerce callback for the attribute, pass VI_NULL for the Coerce Callback parameter.

If you want to specify your own callback function but you want to use the default coerce callback within your function, you can call Ivi_DefaultCoerceCallbackViReal64.
Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle from Ivi_SpecificDriver identifies a particular I
attributeID	ViAttr	Pass the ID of an attri parameter.
		The include file for a s driver defines constar the user-accessible at to the driver. This inclu that the IVI engine det that the instrument cla attributes that are spe particular instrument. constant name begins PREFIX_ATTR_, wher specific instrument pre
		For each IVI engine a specific driver include same constant name ivi.h, except that the s prefix replaces the IVI example, ivi.h defines IVI_ATTR_CACHE, ar include file, fl45.h, def constant name:
		#define FL45_ATTR_ IVI_ATTR_CACHE
		For each instrument c specific driver include same constant name instrument class inclu the specific instrumen

the class prefix. For e class include file, ividi IVIDMM_ATTR_RAN(defines the following c

#define FL45_ATTR_ IVIDMM_ATTR_RA

For each specific instr the specific driver incl constant name and as is an offset from IVI_SPECIFIC_PUBLI For example, fl45.h de constant name:

#define
FL45_ATTR_HOLD_
(IVI_SPECIFIC_PUB
+ 3L)

For each attribute that instrument driver, the source file defines a c assigns a value that is IVI_SPECIFIC_PRIVAT For example, hp34401 following constant nar

#define HP34401A_ATTR_TI (IVI_SPECIFIC_PRI^v + 1L)

coerceCallback CoerceAttrViReal64_CallbackPtr Specify the coerce ca want the IVI engine to attempt to set the attri value.

The function must have prototype:

ViStatus _VI_FUNC Ca

vi,ViConstString repCaı ViAttr attributeId, ViReal64 value, ViReal64 *coercedValue

Note If you war IVI Specific Driv dialog box to de instrument drive retain the paran shown in the pro callback.

If you do not want to L callback function, pas:

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCoerceCallbackViSession

Usage

ViStatus Ivi_SetAttrCoerceCallbackViSession(ViSession vi, ViAttr attributeID, CoerceAttrViSession_CallbackPtr coerceCallback);

Purpose

This function sets the coerce callback function for a ViSession attribute. The IVI engine calls the coerce callback function when you attempt to set the attribute to a new value. The job of the coerce callback is to convert the value you specify into the value to send to the instrument. The IVI engine invokes the coerce callback after it invokes the check callback.

Parameters

Туре	Description
ViSession	The ViSession hand from Ivi_SpecificDrive identifies a particular
ViAttr	 Itom IVI_Specific DIIVE identifies a particular Pass the ID of an attiparameter. The include file for a driver defines constating the user-accessible a to the driver. This ince that the IVI engine det that the IVI engine det that the instrument clattributes that are specific ular instrument. constant name begin PREFIX_ATTR_, whe specific instrument pi For each IVI engine a specific driver include same constant name ivi.h, except that the prefix replaces the IV example ivi h define
	IVI_ATTR_CACHE, <i>e</i> include file, fl45.h, de constant name:
	#define FL45_ATTR IVI_ATTR_CACHE
	For each instrument specific driver include same constant name instrument class inclu the specific instrume
	Type ViSession ViAttr

the class prefix. For e class include file, ivic IVIDMM_ATTR_RAN defines the following

#define FL45_ATTR IVIDMM_ATTR_R/

For each specific insi the specific driver inc constant name and a is an offset from IVI_SPECIFIC_PUBL For example, fl45.h d constant name:

#define FL45_ATTR_HOLD (IVI_SPECIFIC_PU + 3L)

For each attribute tha instrument driver, the source file defines a assigns a value that i IVI_SPECIFIC_PRIVA For example, hp3440 following constant na

#define HP34401A_ATTR_T (IVI_SPECIFIC_PRI + 1L)

coerceCallback CoerceAttrViSession_CallbackPtr Specify the coerce ca want the IVI engine to attempt to set the attivalue.

The function must ha prototype:

ViStatus _VI_FUNC C

vi,ViConstString repCa ViAttr attributeId, ViSession value, ViSession *coercedVal

Note If you wa IVI Specific Dri dialog box to de instrument driv retain the parae shown in the pe callback.

If you do not want to callback function, pas

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCoerceCallbackViString

Usage

ViStatus Ivi_SetAttrCoerceCallbackViString(ViSession vi, ViAttr attributeID, CoerceAttrViString_CallbackPtr coerceCallback);

Purpose

This function sets the coerce callback function for a ViString attribute. The IVI engine calls the coerce callback function when you attempt to set the attribute to a new value. The job of the coerce callback is to convert the value you specify into the value to send to the instrument. The IVI engine invokes the coerce callback after it invokes the check callback.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle t from Ivi_SpecificDriverN identifies a particular IV
attributeID	ViAttr	Pass the ID of an attrib parameter.
		The include file for a sp driver defines constant the user-accessible attu- to the driver. This inclue that the IVI engine define that the instrument class attributes that are spec particular instrument. E constant name begins of PREFIX_ATTR_, where specific instrument pref
		For each IVI engine att specific driver include f same constant name th ivi.h, except that the sp prefix replaces the IVI µ example, ivi.h defines IVI_ATTR_CACHE, and include file, fl45.h, defir constant name:
		#define FL45_ATTR_C IVI_ATTR_CACHE
		For each instrument cla specific driver include f same constant name th instrument class includ the specific instrument

the class prefix. For exiclass include file, ividm IVIDMM_ATTR_RANG defines the following cc

#define FL45_ATTR_F IVIDMM_ATTR_RAN

For each specific instru the specific driver inclu constant name and ass is an offset from IVI_SPECIFIC_PUBLIC For example, fl45.h def constant name:

#define FL45_ATTR_HOLD_T (IVI_SPECIFIC_PUBL + 3L)

For each attribute that i instrument driver, the ir source file defines a co assigns a value that is IVI_SPECIFIC_PRIVAT For example, hp34401a. following constant nam

#define HP34401A_ATTR_TR (IVI_SPECIFIC_PRIV/ + 1L)

coerceCallback CoerceAttrViString_CallbackPtr Specify the coerce call want the IVI engine to i attempt to set the attrib value.

The function must have prototype:

ViStatus _VI_FUNC Call

vi,ViConstString repCapl ViAttr attributeId, const ViConstString value

Unlike the coerce callbathe other data types, you the coerced value to the the last parameter. Inst the coerced value by partice of the coerced value by partice. SetValInStringCallback function.

Note If you want IVI Specific Drive dialog box to dev instrument driver retain the parame shown in the prot callback.

If you do not want to us callback function, pass

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCompareCallbackViAddr

Usage

ViStatus Ivi_SetAttrCompareCallbackViAddr(ViSession vi, ViAttr attributeID, CompareAttrViAddr_CallbackPtr compareCallback);

Purpose

This function sets the compare callback function for a ViAddr attribute. The IVI engine invokes the compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

A compare callback is useful when the instrument can return several values which you consider to have the same meaning, and you do not want to coerce the instrument value in your read callback.

If you do not call this function or you pass VI_NULL for the Compare Callback parameter, the IVI engine makes the comparison based on strict equality.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession hand from Ivi_SpecificDrividentifies a particula
attributeID	ViAttr	Pass the ID of an at parameter.
		The include file for <i>a</i> driver defines constatthe user-accessible to the driver. This in that the IVI engine d that the instrument of attributes that are sparticular instrumen constant name begin PREFIX_ATTR_, wh specific instrument prefix replaces the I' example, ivi.h define IVI_ATTR_CACHE, include file, fl45.h, d constant name:
		#define FL45_ATTI IVI_ATTR_CACHI
		For each instrument specific driver incluc same constant nami instrument class inc the specific instrume

the class prefix. For class include file, ivi IVIDMM_ATTR_RA defines the following

#define FL45_ATTI IVIDMM_ATTR_R For each specific ins the specific driver in

constant name and is an offset from IVI_SPECIFIC_PUBI For example, fl45.h constant name:

#define FL45_ATTR_HOLI (IVI_SPECIFIC_PL + 3L)

For each attribute th instrument driver, th source file defines a assigns a value that IVI_SPECIFIC_PRIV For example, hp344(following constant n

#define HP34401A_ATTR_ (IVI_SPECIFIC_PF + 1L)

compareCallback CompareAttrViAddr_CallbackPtr Specify the compare you want the IVI enc compare a cache va from the instrument you want to set the a

The function must h prototype:

ViStatus _VI_FUNC (vi,ViConstString repC ViAttr attributeId, ViAddr coercedNewV ViAddr cacheValue, ViInt32 *result);

Set *result to a zero and cacheValue are set *result to a non-z

If you want to use th Driver Attributes dia your instrument driver retain the parameter the prototype for the

If you do not want to callback function, pa

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCompareCallbackViBoolean

Usage

ViStatus Ivi_SetAttrCompareCallbackViBoolean(ViSession vi, ViAttr attributeID, CompareAttrViBoolean_CallbackPtr compareCallback);

Purpose

This function sets the compare callback function for a ViBoolean attribute. The IVI engine invokes the compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

A compare callback is useful when the instrument can return several values which you consider to have the same meaning, and you do not want to coerce the instrument value in your read callback.

If you do not call this function or you pass VI_NULL for the Compare Callback parameter, the IVI engine makes the comparison based on strict equality.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession h from Ivi_Specific identifies a partic
attributeID	ViAttr	Pass the ID of ai parameter. The include file f driver defines co the user-accessi to the driver. This that the IVI engir that the IVI engir that the instrume attributes that ar particular instrun
		Constant name b PREFIX_ATTR_, specific instrume For each IVI eng specific driver in same constant n ivi.h, except that prefix replaces th example, ivi.h de IVI_ATTR_CACH include file, fl45.l constant name: #define FL45_A IVI_ATTR_CACH For each instrum specific driver in same constant n instrument class the specific instru

the class prefix. class include file IVIDMM ATTR defines the follov #define FL45 A IVIDMM ATTI For each specific the specific drive constant name a is an offset from IVI SPECIFIC P For example, fl4 constant name: #define FL45 ATTR H (IVI SPECIFIC + 3L) For each attribut instrument driver source file defin€ assigns a value IVI SPECIFIC P For example, hpl following constai #define HP34401A AT. (IVI_SPECIFIC + 1L) compareCallback CompareAttrViBoolean CallbackPtr Specify the com you want the IVI compare a cach from the instrum you want to set t The function mu: prototype:

ViStatus _VI_FUP vi,ViConstString r ViAttr attributeId, ViBoolean coerce ViBoolean cacheV ViInt32 *result);

Set *result to a z coercedNewValu equal. Otherwise zero value.

Note If yo IVI Specific dialog box instrument retain the p shown in the callback.

If you do not war callback function

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCompareCallbackViInt32

Usage

ViStatus Ivi_SetAttrCompareCallbackViInt32(ViSession vi, ViAttr attributeID, CompareAttrViInt32_CallbackPtr compareCallback);

Purpose

This function sets the compare callback function for a VIInt32 attribute. The IVI engine invokes the compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

A compare callback is useful when the instrument can return several values which you consider to have the same meaning, and you do not want to coerce the instrument value in your read callback.

If you do not call this function or you pass VI_NULL for the Compare Callback parameter, the IVI engine makes the comparison based on strict equality.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession hanc from Ivi_SpecificDriv identifies a particula
attributeID	ViAttr	Pass the ID of an at parameter.
		The include file for a driver defines const the user-accessible to the driver. This in that the IVI engine c that the instrument (attributes that are s particular instrumen constant name begi PREFIX_ATTR_, wh specific instrument [
		For each IVI engine specific driver incluc same constant nam ivi.h, except that the prefix replaces the I example, ivi.h define IVI_ATTR_CACHE, include file, fl45.h, d constant name:
		#define FL45_ATT] IVI_ATTR_CACHI
		For each instrument specific driver incluc same constant nam instrument class inc the specific instrume

the class prefix. For class include file, ivi IVIDMM ATTR RA defines the following #define FL45 ATT] IVIDMM_ATTR_R For each specific in: the specific driver in constant name and is an offset from IVI_SPECIFIC_PUB For example, fl45.h constant name: #define FL45 ATTR HOLI (IVI SPECIFIC PL + 3L) For each attribute th instrument driver, th source file defines a assigns a value that IVI SPECIFIC PRIV For example, hp344 following constant n #define HP34401A ATTR (IVI_SPECIFIC_PF + 1L) compareCallback CompareAttrViInt32 CallbackPtr Specify the compare you want the IVI eng compare a cache va from the instrument you want to set the The function must h prototype:

ViStatus _VI_FUNC (vi,ViConstString repC ViAttr attributeId, ViInt32 coercedNewV ViInt32 cacheValue, ViInt32 *result);

Set *result to a zero coercedNewValue a equal. Otherwise, se zero value.

Note If you w IVI Specific D dialog box to (instrument dri retain the para shown in the p callback.

If you do not want to callback function, pa

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCompareCallbackViInt64

Usage

ViStatus Ivi_SetAttrCompareCallbackViInt64(ViSession vi, ViAttr attributeID, CompareAttrViInt64_CallbackPtr compareCallback);

Purpose

This function sets the compare callback function for a VIInt64 attribute. The IVI engine invokes the compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

A compare callback is useful when the instrument can return several values which you consider to have the same meaning, and you do not want to coerce the instrument value in your read callback.

If you do not call this function or you pass VI_NULL for the Compare Callback parameter, the IVI engine makes the comparison based on strict equality.
Parameters

Name	Туре	Description
vi	ViSession	The ViSession hanc from Ivi_SpecificDriv identifies a particula
attributeID	ViAttr	Pass the ID of an at parameter.
		The include file for a driver defines const the user-accessible to the driver. This in that the IVI engine c that the instrument (attributes that are s particular instrumen constant name begi PREFIX_ATTR_, wh specific instrument [
		For each IVI engine specific driver incluc same constant nam ivi.h, except that the prefix replaces the I example, ivi.h define IVI_ATTR_CACHE, include file, fl45.h, d constant name:
		#define FL45_ATT] IVI_ATTR_CACHI
		For each instrument specific driver incluc same constant nam instrument class inc the specific instrume

the class prefix. For class include file, ivi IVIDMM ATTR RA defines the following #define FL45 ATT] IVIDMM_ATTR_R For each specific in: the specific driver in constant name and is an offset from IVI_SPECIFIC_PUB For example, fl45.h constant name: #define FL45 ATTR HOLI (IVI SPECIFIC PL + 3L) For each attribute th instrument driver, th source file defines a assigns a value that IVI SPECIFIC PRIV For example, hp344 following constant n #define HP34401A ATTR (IVI_SPECIFIC_PF + 1L) compareCallback CompareAttrViInt64 CallbackPtr Specify the compare you want the IVI eng compare a cache va from the instrument you want to set the The function must h prototype:

ViStatus _VI_FUNC (vi,ViConstString repC ViAttr attributeId, ViInt32 coercedNewV ViInt32 cacheValue, ViInt32 *result);

Set *result to a zero coercedNewValue a equal. Otherwise, se zero value.

Note If you w IVI Specific D dialog box to (instrument dri retain the para shown in the p callback.

If you do not want to callback function, pa

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCompareCallbackViReal64

Usage

ViStatus Ivi_SetAttrCompareCallbackViReal64(ViSession vi, ViAttr attributeID, CompareAttrViReal64_CallbackPtr compareCallback);

Purpose

This function sets the compare callback function for a ViReal64 attribute. The IVI engine invokes the compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

A compare callback is useful when the instrument can return several values which you consider to have the same meaning, and you do not want to coerce the instrument value in your read callback.

When you create an attribute with Ivi_AddAttributeViReal64, the IVI engine automatically installs a default compare callback. The default callback uses the degree of precision you specify in the Compare Precision parameter to Ivi_AddAttributeViReal64. The IVI engine installs the default compare callback rather than comparing based on strict equality because of differences between computer and instrument floating point representations.

If you want to compare based on strict equality, pass VI_NULL for the Compare Callback parameter. If you set the callback to VI_NULL and subsequently call Ivi_SetAttrComparePrecision on the attribute, the IVI engine reinstalls its default compare callback.

If you want to specify your own callback function but you want to use the default compare callback within your function, you can call Ivi_DefaultCompareCallbackViReal64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession ha from Ivi_SpecificD identifies a particı
attributeID	ViAttr	Pass the ID of an parameter.
		The include file fo driver defines con the user-accessib to the driver. This that the IVI engine that the instrumer attributes that are particular instrumer constant name be PREFIX_ATTR_, v specific instrumer
		For each IVI engin specific driver incl same constant na ivi.h, except that t prefix replaces the example, ivi.h def IVI_ATTR_CACH include file, fl45.h constant name:
		#define FL45_AT IVI_ATTR_CAC
		For each instrume specific driver incl same constant na instrument class i the specific instru

```
the class prefix. F
                                                       class include file,
                                                       IVIDMM ATTR F
                                                       defines the follow
                                                        #define FL45 AT
                                                        IVIDMM_ATTR
                                                       For each specific
                                                       the specific driver
                                                       constant name ar
                                                       is an offset from
                                                       IVI_SPECIFIC_PU
                                                       For example, fl45
                                                       constant name:
                                                         #define
                                                        FL45 ATTR HC
                                                        (IVI_SPECIFIC_
                                                         + 3L)
                                                       For each attribute
                                                       instrument driver.
                                                       source file define:
                                                       assigns a value th
                                                       IVI SPECIFIC PR
                                                       For example, hp3<sup>4</sup>
                                                       following constant
                                                         #define
                                                         HP34401A ATT
                                                         (IVI_SPECIFIC_
                                                         + 1L)
compareCallback CompareAttrViReal64 CallbackPtr
                                                       Specify the comp
                                                       you want the IVI є
                                                       compare a cache
                                                       from the instrume
                                                       you want to set th
                                                       The function must
                                                       prototype:
```

ViStatus _VI_FUN vi,ViConstString re ViAttr attributeId, ViReal64 coercedN ViReal64 cacheValı ViInt32 *result);

Set *result to a ze coercedNewValue equal. Otherwise, zero value.

Note If you IVI Specific dialog box t instrument (retain the pa shown in the callback.

If you do not want callback function,

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCompareCallbackViSession

Usage

ViStatus Ivi_SetAttrCompareCallbackViSession(ViSession vi, ViAttr attributeID, CompareAttrViSession_CallbackPtr compareCallback);

Purpose

This function sets the compare callback function for a ViSession attribute. The IVI engine invokes the compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

A compare callback is useful when the instrument can return several values which you consider to have the same meaning, and you do not want to coerce the instrument value in your read callback.

If you do not call this function or you pass VI_NULL for the Compare Callback parameter, the IVI engine makes the comparison based on strict equality.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession had from Ivi_Specificl identifies a partic
attributeID	ViAttr	Pass the ID of ar parameter.
		The include file f driver defines co the user-accessil to the driver. This that the IVI engir that the IVI engir that the instrume attributes that are particular instrum constant name b PREFIX_ATTR_, specific instrume
		For each IVI eng specific driver ind same constant n ivi.h, except that prefix replaces th example, ivi.h de IVI_ATTR_CACH include file, fl45.h constant name:
		#define FL45_A IVI_ATTR_CA(
		For each instrum specific driver inc same constant n instrument class the specific instru

the class prefix. I class include file IVIDMM ATTR defines the follov #define FL45 A IVIDMM ATTE For each specific the specific drive constant name a is an offset from IVI_SPECIFIC_P For example, fl4! constant name: #define FL45 ATTR HO (IVI_SPECIFIC + 3L) For each attribut instrument driver source file define assigns a value t IVI SPECIFIC P For example, hp? following constar #define HP34401A AT7 (IVI_SPECIFIC + 1L) compareCallback CompareAttrViSession CallbackPtr Specify the comp you want the IVI compare a cache from the instrum you want to set t The function mus prototype:

ViStatus _VI_FUN vi,ViConstString r ViAttr attributeId, ViSession coerced ViSession cacheVa ViInt32 *result);

Set *result to a z coercedNewValu equal. Otherwise zero value.

Note If yo IVI Specific dialog box instrument retain the shown in th callback.

If you do not war callback function

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrCompareCallbackViString

Usage

ViStatus Ivi_SetAttrCompareCallbackViString(ViSession vi, ViAttr attributeID, CompareAttrViString_CallbackPtr compareCallback);

Purpose

This function sets the compare callback function for a ViString attribute. The IVI engine invokes the compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the compare callback determines that the two values are equal, the IVI engine does not call the write callback for the attribute.

A compare callback is useful when the instrument can return several values which you consider to have the same meaning, and you do not want to coerce the instrument value in your read callback.

If you do not call this function or you pass VI_NULL for the Compare Callback parameter, the IVI engine makes the comparison based on strict equality.

Parameters

Туре	Description
ViSession	The ViSession han from Ivi_SpecificDr identifies a particul
ViAttr	Pass the ID of an <i>a</i> parameter.
	The include file for driver defines cons the user-accessible to the driver. This in that the IVI engine that the instrument attributes that are s particular instrument constant name beg PREFIX_ATTR_, w specific instrument
	For each IVI engine specific driver inclu same constant nan ivi.h, except that th prefix replaces the example, ivi.h defir IVI_ATTR_CACHE include file, fl45.h, (constant name:
	#define FL45_AT1 IVI_ATTR_CACH
	For each instrumer specific driver inclu same constant nan instrument class in the specific instrum
	Type ViSession

the class prefix. Fo class include file, iv IVIDMM ATTR RA defines the followir #define FL45 AT1 IVIDMM_ATTR_] For each specific ir the specific driver i constant name and is an offset from IVI_SPECIFIC_PUE For example, fl45.h constant name: #define FL45 ATTR HOL (IVI SPECIFIC P + 3L) For each attribute t instrument driver, t source file defines assigns a value the IVI SPECIFIC PRI For example, hp344 following constant #define HP34401A_ATTR (IVI_SPECIFIC_P + 1L) compareCallback CompareAttrViString CallbackPtr Specify the compa you want the IVI er compare a cache v from the instrumen you want to set the The function must prototype:

ViStatus _VI_FUNC vi,ViConstString rep ViAttr attributeId, ViConstString coerce ViConstString cache ViInt32 *result);

Set *result to a zero coercedNewValue equal. Otherwise, s zero value.

Note If you N IVI Specific E dialog box to instrument du retain the pau shown in the callback.

If you do not want 1 callback function, p

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrComparePrecision

Usage

ViStatus Ivi_SetAttrComparePrecision(ViSession vi, ViAttr attributeID, ViInt32 comparePrecision);

Purpose

This function changes the degree of decimal precision the default IVI compare callback uses for a specific attribute.

This function is useful only for ViReal64 attributes. You set the initial comparison precision level for an attribute as a parameter to the Ivi_AddAttributeViReal64 function.

Unless you call Ivi_SetAttrCompareCallbackViReal64 to install your own compare callback function, the IVI engine invokes the default compare callback when comparing cache values it obtains from the instrument against new values you set the attribute to. If the values are equal within the degree of precision you specify, the IVI engine does not call the write callback for the attribute.

The IVI engine uses this method instead of strict equality because of differences between computer and instrument floating point representations.

If the compare callback for the attribute is currently VI_NULL, this function installs the default IVI compare callback.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID ViAttr	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines

	IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:
	#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE
	For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:
	#define FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)
	For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:
	#define HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)
comparePrecision Vilnt32	The degree of precision you want the default IVI compare callback to use for this attribute.
	The value for this parameter is in terms of decimal digits of precision. The higher the value, the closer the two values must be for the default compare callback to consider them equal.
	Valid Range: 0, or 1 to 14

If you pass 0, the function sets the precision to the default IVI for this value, which is 14.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttributeFlags

Usage

ViStatus Ivi_SetAttributeFlags(ViSession vi, ViAttr attributeID, IviAttrFlags flags);

Purpose

This function sets the flags of an attribute to new values. It always sets all of the flags. If you want to change one flag, use Ivi_GetAttributeFlags to obtain the current values of all the flags, modify the bit for the flag you want to change, and then call Ivi_SetAttributeFlags.

You cannot modify the value of the IVI_VAL_MULTI_CHANNEL flag.

Parameters

Name	Туре	Description
vi ViSession	ViSession	Returns a ViSession handle that you use to identify the session in subsequent function calls.
		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.
		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 same session in multiple execution threads. You can use functions Ivi_LockSession and Ivi_UnlockSession to protect sections of code that require exclusive access to the resource.
attributeID ViAttr	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user- accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define FL45_ATTR_HOLD_THRESHOLD \
(IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

flags IviAttrFlags Specify the new values of the flags for the attribute. You express the flags as bits. You cannot modify the value of the IVI_VAL_MULTI_CHANNEL flag.

This function always sets all of the flags. If you want to change one flag, use

		Ivi_GetAttributeFlags to obtain the current values of all of the flags, modify the bit of the flag you want to change, and then call Ivi_SetAttributeFlags.
		The code in the following example changes the IVI_VAL_NEVER_CACHE flag from 1 to 0.
		IviAttrFlags oldFlags, newFlags;
		Ivi_GetAttributeFlags (vi, attributeID, &oldFlags);newFlags = oldFlags & ~IVI_VAL_NEVER_CACHE; Ivi_SetAttributeFlags (vi, attributeID, newFlags);
		Valid Values:
		Bit Value Flag
		0
1	0x0002	IVI_VAL_NOT_READABLE
2	0x0004	IVI_VAL_NOT_WRITABLE
3	0x0008	IVI_VAL_NOT_USER_READABLE
4	0x0010	IVI_VAL_NOT_USER_WRITABLE
5	0x0020	IVI_VAL_NEVER_CACHE
6	0x0040	IVI_VAL_ALWAYS_CACHE
10	0x0400	IVI_VAL_MULTI_CHANNEL
11	0x0800	IVI_VAL_COERCEABLE_ONLY_BY_INSTR
12	0x1000	IVI_VAL_WAIT_FOR_OPC_BEFORE_READS
13	0x2000	IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES
14	0x4000	IVI_VAL_USE_CALLBACKS_FOR_SIMULATION
15	0x8000	IVI_VAL_DONT_CHECK_STATUS

IVI_VAL_HIDDEN is 0x0018, the combination of

IVI_VAL_NOT_USER_READABLE and

IVI_VAL_NOT_USER_WRITABLE.

See the control help for the Flags parameter to the Ivi_AddAttribute functions for detailed information on each flag.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttributeViAddr

Usage

ViStatus Ivi_SetAttributeViAddr(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViAddr attributeValue[]);

Purpose

This function sets the ViAddr attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If IVI_ATTR_RANGE_CHECK is enabled for the session and you provide a check callback for the attribute, the function invokes the check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. If you provide a coerce callback, the function invokes the callback to coerce the value.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. When the cache value is a value the IVI engine obtained by querying the instrument and you provide a compare callback for the attribute, the function invokes the compare callback. Otherwise, the function makes the comparison based on strict equality.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.
- **Note** If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option Flags parameter, or if the IVI_ATTR_SIMULATE attribute is
enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>
		If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <enter>.</enter>
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViAddr Pass the value to which you want to set the attribute. If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can bring up a list of them on this control

by pressing ENTER. Select a value by doubleclicking on it or by highlighting it and then pressing ENTER. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttributeViBoolean

Usage

ViStatus Ivi_SetAttributeViBoolean(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViBoolean attributeValue);

Purpose

This function sets the ViBoolean attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If IVI_ATTR_RANGE_CHECK is enabled for the session and you provide a check callback for the attribute, the function invokes the check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. Normally, it uses the default IVI coerce callback for ViBoolean attributes.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. When the cache value is a value the IVI engine obtained by querying the instrument and you provide a compare callback for the attribute, the function invokes the compare callback. Otherwise, the function makes the comparison based on strict equality.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.
- **Note** If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option Flags parameter, or if the IVI_ATTR_SIMULATE attribute is

enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>

If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <ENTER>.

optionFlags

Vilnt32

³² Use this parameter to request special behavior. In most cases, you pass 0.

You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViBoolean Pass the attribute

enter the class or driver at press <(to toggle control t manual so you c the ID co If you wa enter the an IVI at then you this cont ring con When yo on this r control c

<ENTEF <SPACE or <CTF DOWN> dialog b appears containii hierarch of the at the IVI e defines. dialog sl help text attribute you sele attribute double-c on it or k highlight and ther pressing <ENTEF dialog disappe your chc appears function control. If this fu a SetAtt **CheckA** function only attr appear (the list b you sele read-onl attribute

message appears A ring cc the top c dialog b you to se attribute the attrik that hav types co with this If you ch see all I attribute data typ appear t right of t attribute in the lis The data that are consiste this func dim. If ye an attrib type that LabWinc transfers the func panel fo correspo function consiste the data If the att this ring has nam constant

valid val can brin list of the moving 1 Attribute control a pressing <ENTEF

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttributeViInt32

Usage

ViStatus Ivi_SetAttributeViInt32(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViInt32 attributeValue);

Purpose

This function sets the ViInt32 attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify if IVI_ATTR_RANGE_CHECK is enabled for the session. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. If you provide a coerce callback, the function invokes the callback to coerce the value. If you do not provide a coerce callback but you provide a coerced range table directly or through a range table callback, the function invokes the default IVI coerce callback to coerce the value.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. When the cache value is a value the IVI engine obtained by querying the instrument and you provide a compare callback for the attribute, the function invokes the compare callback. Otherwise, the function makes the comparison based on strict equality.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS

attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

 $\overline{\mathbb{N}}$

Note If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option Flags parameter, or if the IVI_ATTR_SIMULATE attribute is enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Parameters		
Name	Туре	Description
vi	ViSession	Returns a ViSession handle that you use to identify the session in subsequent function calls.
		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.
		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 same session in multiple execution threads. You can use functions Ivi_LockSession and Ivi_UnlockSession to protect sections of code that require exclusive access to the resource.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle

this ring control to the manual input box so you can type the ID constant.

If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <ENTER>, <SPACEBAR>, or <CTRL-DOWN>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <ENTER>, the dialog disappears and your choice appears in this function panel control.

If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <ENTER>.

optionFlagsViInt32Use this parameter to request special
behavior. In most cases, you pass 0.You can specify individual bits to

request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

Vilnt32 Pass the value to which you want to set the attribute. If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can bring up a list of them on this control

by pressing <ENTER>. Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttributeViInt64

Usage

ViStatus Ivi_SetAttributeViInt64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViInt64 attributeValue);

Purpose

This function sets the ViInt64 attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify if IVI_ATTR_RANGE_CHECK is enabled for the session. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. If you provide a coerce callback, the function invokes the callback to coerce the value. If you do not provide a coerce callback but you provide a coerced range table directly or through a range table callback, the function invokes the default IVI coerce callback to coerce the value.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. When the cache value is a value the IVI engine obtained by querying the instrument and you provide a compare callback for the attribute, the function invokes the compare callback. Otherwise, the function makes the comparison based on strict equality.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS

attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

 $\overline{\mathbb{N}}$

Note If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option Flags parameter, or if the IVI_ATTR_SIMULATE attribute is enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Parameters		
Name	Туре	Description
vi	ViSession	Returns a ViSession handle that you use to identify the session in subsequent function calls.
		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.
		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 same session in multiple execution threads. You can use functions Ivi_LockSession and Ivi_UnlockSession to protect sections of code that require exclusive access to the resource.
repeatedCapability \	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument

driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.

For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE.
		For example, fl45.h defines the following constant name:
		#define FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)
		For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:
		#define HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)
optionFlags	Vilnt32	Use this parameter to request special behavior. In most cases, you pass 0.
		You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

Vilnt64 Pass the value to which you want to set the attribute. If the attribute currently showing in the Attribute ID ring control has named constants as valid values, you can bring up a list of them on this control

by pressing <ENTER>. Select a value by doubleclicking on it or by highlighting it and then pressing <ENTER>. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttributeViReal64

Usage

ViStatus Ivi_SetAttributeViReal64(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViReal64 attributeValue[]);

Purpose

This function sets the ViReal64 attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify if IVI_ATTR_RANGE_CHECK is enabled for the session. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. If you provide a coerce callback, the function invokes the callback to coerce the value. If you do not provide a coerce callback but you provide a coerced range table directly or through a range table callback, the function invokes the default IVI coerce callback to coerce the value.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. The method it uses depends on the source of the cache value. If the cache contains a value you previously sent to the instrument using this function, the function compares the two values using strict equality. If the cache contains a value you obtained from the instrument using Ivi_GetAttributeViReal64, the function invokes the compare callback you provide for the attribute or the default IVI compare callback, which uses the comparison precision you specify when you create the attribute.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for

the attribute, the function invokes the operation complete (OPC) callback you provide for the session.

- 7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.
- Note If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option Flags parameter, or if the IVI_ATTR_SIMULATE attribute is enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
repeatedCapability	ViChar[]	If the attribute you specify is based on a repeated capability, pass a repeated capability identifier. You can pass one of the identifiers strings that the specific instrument driver defines, or a virtual name the end-user defines in the configuration file.
		If the attribute you specify is not based on a repeated capability, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this ring control or press <enter>, <spacebar>, or <ctrl-down>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <enter>, the dialog disappears and your choice appears in this function panel control.</enter></ctrl-down></spacebar></enter>

If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <ENTER>.

optionFlags

Vilnt32

³² Use this parameter to request special behavior. In most cases, you pass 0.

You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViReal64 Pass the value to which you want to se the attribute. If the attribute currently showing i the Attribute ring contr has name constants as valid values, y can bring up a list c them on this contr

by pressing <ENTER: Select a value by doubleclicking o it or by highlightii it and the pressing <ENTER: Some of the value might not be valid dependin on the current settings c the instrumer session.

Return Value

This function sets the ViReal64 attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify if IVI_ATTR_RANGE_CHECK is enabled for the session. If you provide a check callback, the function invokes the callback to validate the value. If you do not provide a check callback but you provide a range table or a range table callback, the function invokes the default IVI check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. If you provide a coerce callback, the function invokes the callback to coerce the value. If you do not provide a coerce callback but you provide a coerced range table directly or through a range table callback, the function invokes the default IVI coerce callback to coerce the value.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. The method it uses depends on the source of the cache value. If the cache contains a value you previously sent to the instrument using this function, the function compares the two values using strict equality. If the cache contains a value you obtained from the instrument using Ivi_GetAttributeViReal64, the function invokes the compare callback you provide for the attribute or the default IVI compare callback, which uses the comparison precision you specify when you create the attribute.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for

the attribute, the function invokes the operation complete (OPC) callback you provide for the session.

7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.

Note If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option Flags parameter, or if the IVI_ATTR_SIMULATE attribute is enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Related Topic

IVI Status Codes

Ivi_SetAttributeViSession

Usage

ViStatus Ivi_SetAttributeViSession(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViSession attributeValue);

Purpose

This function sets the ViSession attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If IVI_ATTR_RANGE_CHECK is enabled for the session and you provide a check callback for the attribute, the function invokes the check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. If you provide a coerce callback, the function invokes the callback to coerce the value. Generally, ViSession attributes do not have coerce callbacks.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. When the cache value is a value the IVI engine obtained by querying the instrument and you provide a compare callback for the attribute, the function invokes the compare callback. Otherwise, the function makes the comparison based on strict equality.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.
- **Note** If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option

Flags parameter, or if the IVI_ATTR_SIMULATE attribute is enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Parameters		
Name	Туре	Description
vi	ViSession	Returns a ViSession handle that you use to identify the session in subsequent function calls.
		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.
		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 same session in multiple execution threads. You can use functions Ivi_LockSession and Ivi_UnlockSession to protect sections of code that require exclusive access to the resource.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this

ring control or press <ENTER>, <SPACEBAR>, or <CTRL-DOWN>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <ENTER>, the dialog disappears and your choice appears in this function panel control.

If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <ENTER>.

optionFlags

Vilnt32

Use this parameter to request special behavior. In most cases, you pass 0.

You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.

IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

value to which yc want to s the attribute If the attribute

ViSession Pass the

If the attribute currently showing the Attribute ring cont has nam constant as valid values, y can bring up a list them on this cont

by pressing <ENTEF Select a value by doubleclicking (it or by highlight it and the pressing <ENTEF Some of the value might nc be valid dependii on the current settings the instrume session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttributeViString

Usage

ViStatus Ivi_SetAttributeViString(ViSession vi, ViChar repeatedCapability[], ViAttr attributeID, ViInt32 optionFlags, ViChar attributeValue[]);

Purpose

This function sets the ViString attribute you specify to a new value. Depending on the configuration of the attribute, the function performs the following actions:

- 1. Checks whether the attribute is writable. If not, the function returns an error.
- 2. Validates the value you specify. If IVI_ATTR_RANGE_CHECK is enabled for the session and you provide a check callback for the attribute, the function invokes the check callback to validate the value. If the value is invalid, the function returns an error.
- 3. Coerces the value you specify into a canonical value the instrument accepts. If you provide a coerce callback, the function invokes the callback to coerce the value.
- 4. Compares the new value with the current cache value for the attribute to see if they are equal. When the cache value is a value the IVI engine obtained by querying the instrument and you provide a compare callback for the attribute, the function invokes the compare callback. Otherwise, the function makes the comparison based on strict equality.
- 5. If the new value is not equal to the cache value or the cache value is invalid, the function invokes the write callback for the attribute. The write callback might perform I/O to send the value to the instrument. The IVI engine stores the new value in the cache. If the function coerces the value, the function caches the coerced value rather than the value you pass. The function allocates a copy of the string to keep in the cache.
- 6. If the IVI_VAL_WAIT_FOR_OPC_AFTER_WRITES flag is set for the attribute, the function invokes the operation complete (OPC) callback you provide for the session.
- 7. If you set the IVI_VAL_DIRECT_USER_CALL bit in the Option Flags parameter, the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute is enabled, and the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0, then the function invokes the check status callback you provide for the session.
- **Note** If you set the IVI_VAL_SET_CACHE_ONLY bit in the Option

Flags parameter, or if the IVI_ATTR_SIMULATE attribute is enabled and the IVI_VAL_USE_CALLBACKS_FOR_SIMULATION flag for the attribute is 0, the function does not call the write callback, the operation complete callback, or the check status callback. It merely updates the cache value of the attribute.

Parameters		
Name	Туре	Description
vi	ViSession	Returns a ViSession handle that you use to identify the session in subsequent function calls.
		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.
		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 same session in multiple execution threads. You can use functions Ivi_LockSession and Ivi_UnlockSession to protect sections of code that require exclusive access to the resource.
repeatedCapability	ViChar[]	If the attribute is repeated capability- based, specify a particular repeated capability. If the attribute you specify is not repeated capability-based, pass VI_NULL or an empty string.
attributeID	ViAttr	Pass the ID of an attribute.
		If you want to enter the ID of a class or specific driver attribute, press to toggle this ring control to the manual input box so you can type the ID constant.
		If you want to enter the ID of an IVI attribute, then you can use this control as a ring control. When you click on this

ring control or press <ENTER>, <SPACEBAR>, or <CTRL-DOWN>, a dialog box appears containing a hierarchical list of the attributes the IVI engine defines. The dialog shows help text for each attribute. When you select an attribute by double-clicking on it or by highlighting it and then pressing <ENTER>, the dialog disappears and your choice appears in this function panel control.

If this function is a SetAttribute or CheckAttribute function, read-only attributes appear dim in the list box. If you select a read-only an 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 that have data types consistent with this function. If you choose to see all IVI attributes, the data types appear to the right of the attribute names in the list box. The data types that are not consistent with this function 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 the attribute in this ring control has named constants as valid values, you can bring up a list of them by moving to the Attribute Value control and pressing <ENTER>.

optionFlags

Vilnt32

Use this parameter to request special behavior. In most cases, you pass 0.

You can specify individual bits to request specific behavior. Each of the following sections describes one of the bit values.

IVI_VAL_DIRECT_USER_CALL (1<<0)

Use the IVI_VAL_DIRECT_USER_CALL bit when calling this function from the source code for the PREFIX_Set/Get/CheckAttribute functions that your instrument driver exports. Do not use the bit when calling this function from any other context.

If you set this bit, this function checks the IVI_VAL_USER_READABLE or IVI_VAL_USER_WRITABLE flag. If the end-user does not have permission to access the attribute, the function returns an error.

If you set this bit, the function also checks the status of the instrument after invoking the read or write callback for the attribute, but only if the following conditions are true.

- This is a SetAttribute or GetAttribute function.
- The value of the IVI_ATTR_QUERY_INSTRUMENT_STATUS attribute for the session is VI_TRUE.
- The value of the IVI_VAL_DONT_CHECK_STATUS flag for the attribute is 0.

IVI_VAL_SET_CACHE_ONLY (1<<1)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set the value in the attribute cache but you do not want to invoke the write callback for the attribute.

This is useful if one instrument I/O command sets multiple attributes in the instrument. In the write callback function that performs the instrument I/O, after the instrument I/O succeeds, call an Ivi_SetAttribute function for each of the other attributes, with set the IVI_VAL_SET_CACHE_ONLY bit set to 1 in the Option Flags parameter.
IVI_VAL_DONT_MARK_AS_SET_BY_USER (1<<2)

This bit applies only to calls that specific instrument drivers make to the Ivi_SetAttribute functions. Pass 1 for this bit if want to set an attribute value even though the user has not requested you to do so directly through a PREFIX_SetAttribute function call or indirectly through a helper function that sets multiple attributes.

This case occurs very rarely. It affects interchangeability checking in class drivers. To pass interchangeability checking, either all attributes in an extension group must be marked as "set by user" or none of them must be marked as "set by user".

attributeValue

ViChar[Pass the

1

value to which you want to set the attribute. If the attribute currently showing in the Attribute IC ring contro has named constants as valid values, you can bring up a list of them on this control

by pressing ENTER. Select a value by doubleclicking on it or by highlighting it and then pressing ENTER. Some of the values might not be valid depending on the current settings of the instrument session.

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrRangeTableCallback

Usage

ViStatus Ivi_SetAttrRangeTableCallback(ViSession vi, ViAttr attributeID, RangeTable_CallbackPtr rangeTableCallback);

Purpose

This function sets the callback that the IVI engine invokes to obtain a pointer to the range table for an attribute. Although any attribute can have a range table, range tables normally are useful only for ViInt32 or ViReal64 attributes.

When you create a ViInt32 or ViReal64 attribute, you can specify a single range table for the IVI engine to use to validate values for the attribute. Normally, one range table is sufficient. If this is the case, you do not need a range table callback function. By default, the range table callback for each attribute is VI_NULL.

Sometimes, however, you want to use different range tables depending on the current settings of other attributes. In that case, call Ivi_SetAttrRangeTableCallback to install a callback for the IVI engine to invoke. In the callback, you determine which range table you want to use, and you return a pointer to it.

When you specify a non-NULL range table callback for a ViInt32 or ViReal64 attribute, the IVI engine automatically installs its default check and coerce callbacks if these callback are currently VI_NULL.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tha from Ivi_SpecificDriverNet identifies a particular IVI s
attributeID	ViAttr	Pass the ID of an attribut
		The include file for a spea driver defines constant na the user-accessible attrib to the driver. This include that the IVI engine define that the instrument class attributes that are specific particular instrument. Eac constant name begins wi PREFIX_ATTR_, where P specific instrument prefix
		For each IVI engine attrik specific driver include file same constant name that ivi.h, except that the spec prefix replaces the IVI pre example, ivi.h defines IVI_ATTR_CACHE, and t include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include t the specific instrument pr

the class prefix. For exan class include file, ividmm. IVIDMM_ATTR_RANGE, defines the following con:

#define FL45_ATTR_RA IVIDMM_ATTR_RANG

For each specific instrum the specific driver include constant name and assig is an offset from IVI_SPECIFIC_PUBLIC_/ For example, fl45.h define constant name:

#define
FL45_ATTR_HOLD_TH
(IVI_SPECIFIC_PUBLIC
+ 3L)

For each attribute that is instrument driver, the inst source file defines a cons assigns a value that is ar IVI_SPECIFIC_PRIVATE_ For example, hp34401a.c following constant name:

#define HP34401A_ATTR_TRIG (IVI_SPECIFIC_PRIVAT + 1L)

rangeTableCallback RangeTable_CallbackPtr Specify the range table c function you want the IVI invoke to obtain a range t attribute.

The function must have the prototype:

ViStatus _VI_FUNC Callba

vi,ViConstString repCapNa ViAttr attributeId, IviRangeTablePtr *rangeTa If you do not want to use callback function, pass V

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrReadCallbackViAddr

Usage

ViStatus Ivi_SetAttrReadCallbackViAddr(ViSession vi, ViAttr attributeID, ReadAttrViAddr_CallbackPtr readCallback);

Purpose

This function sets the read callback function for a ViAddr attribute. The IVI engine calls the read callback function when you request the current value of the attribute and the cache value is invalid.

If you do not want the IVI engine to invoke a read callback, specify VI_NULL for the Read Callback parameter.

You can set the read callback function when you create the attribute with Ivi_AddAttributeViAddr.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that ye from Ivi_SpecificDriverNew. T identifies a particular IVI ses
attributeID	ViAttr	Pass the ID of an attribute fo parameter.
		The include file for a specific driver defines constant name the user-accessible attributes to the driver. This includes at that the IVI engine defines, a that the instrument class defi attributes that are specific to particular instrument. Each d constant name begins with PREFIX_ATTR_, where PREI specific instrument prefix.
		For each IVI engine attribute specific driver include file use same constant name that ap ivi.h, except that the specific prefix replaces the IVI prefix. example, ivi.h defines IVI_ATTR_CACHE, and the I include file, fl45.h, defines the constant name:
		#define FL45_ATTR_CACH IVI_ATTR_CACHE
		For each instrument class at specific driver include file use same constant name that ap instrument class include file, the specific instrument prefix

the class prefix. For example class include file, ividmm.h, c IVIDMM_ATTR_RANGE, and defines the following constar

#define FL45_ATTR_RANGE

For each specific instrument the specific driver include file constant name and assigns a is an offset from

IVI_SPECIFIC_PUBLIC_ATT For example, fl45.h defines t constant name:

#define

FL45_ATTR_HOLD_THRES (IVI_SPECIFIC_PUBLIC_A' + 3L)

For each attribute that is priv instrument driver, the instrum source file defines a constan assigns a value that is an off IVI_SPECIFIC_PRIVATE_ATT For example, hp34401a.c defi following constant name:

#define

HP34401A_ATTR_TRIGGE (IVI_SPECIFIC_PRIVATE_/ + 1L)

readCallback ReadAttrViAddr_CallbackPtr Specify the read callback fur want the IVI engine to invoke request the current value of 1 attribute.

You must define the read cal function in the source code for specific instrument driver. Th must have the following prote

ViStatus _VI_FUNC Callback(' vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViAddr *value);

Upon entry to the callback, * contains the cache value. Up from the callback, *value mu the actual current value.

Note If you want to us IVI Specific Driver Attri dialog box to develop y instrument driver sourc retain the parameter na shown in the prototype callback.

If you do not want to use a re callback function, pass VI_N

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrReadCallbackViBoolean

Usage

ViStatus Ivi_SetAttrReadCallbackViBoolean(ViSession vi, ViAttr attributeID, ReadAttrViBoolean_CallbackPtr readCallback);

Purpose

This function sets the read callback function for a ViBoolean attribute. The IVI engine calls the read callback function when you request the current value of the attribute and the cache value is invalid.

If you do not want the IVI engine to invoke a read callback, specify VI_NULL for the Read Callback parameter.

You can set the read callback function when you create the attribute with Ivi_AddAttributeViBoolean.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tha from Ivi_SpecificDriverNet identifies a particular IVI s
attributeID	ViAttr	Pass the ID of an attribut parameter.
		The include file for a spee driver defines constant na the user-accessible attrib to the driver. This include that the IVI engine define that the instrument class attributes that are specific particular instrument. Eac constant name begins wi PREFIX_ATTR_, where P specific instrument prefix
		For each IVI engine attrik specific driver include file same constant name that ivi.h, except that the spec prefix replaces the IVI pre example, ivi.h defines IVI_ATTR_CACHE, and t include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include t the specific instrument pr

the class prefix. For exan class include file, ividmm. IVIDMM_ATTR_RANGE, defines the following con:

#define FL45_ATTR_RA IVIDMM_ATTR_RANG

For each specific instrum the specific driver include constant name and assig is an offset from IVI_SPECIFIC_PUBLIC_/ For example, fl45.h define constant name:

#define
FL45_ATTR_HOLD_TH
(IVI_SPECIFIC_PUBLIC
+ 3L)

For each attribute that is instrument driver, the inst source file defines a cons assigns a value that is ar IVI_SPECIFIC_PRIVATE_ For example, hp34401a.c following constant name:

#define HP34401A_ATTR_TRIG (IVI_SPECIFIC_PRIVAT + 1L)

readCallback ReadAttrViBoolean_CallbackPtr Specify the read callback want the IVI engine to inv request the current value attribute. You must define the read

You must define the read function in the source coc specific instrument driver must have the following p

ViStatus _VI_FUNC Callba vi,ViSession io, ViConstString repCapName ViAttr attributeId, ViBoolean *value);

Upon entry to the callbac contains the cache value from the callback, *value the actual current value.

Note If you want to IVI Specific Driver / dialog box to develo instrument driver so retain the paramete shown in the protot callback.

If you do not want to use callback function, pass $\rm V$

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrReadCallbackViInt32

Usage

ViStatus Ivi_SetAttrReadCallbackViInt32(ViSession vi, ViAttr attributeID, ReadAttrViInt32_CallbackPtr readCallback);

Purpose

This function sets the read callback function for a ViInt32 attribute. The IVI engine calls the read callback function when you request the current value of the attribute and the cache value is invalid.

If you do not want the IVI engine to invoke a read callback, specify VI_NULL for the Read Callback parameter.

You can set the read callback function when you create the attribute with Ivi_AddAttributeViInt32.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that ye from Ivi_SpecificDriverNew. 7 identifies a particular IVI ses
attributeID	ViAttr	Pass the ID of an attribute fc parameter.
		The include file for a specific driver defines constant name the user-accessible attribute to the driver. This includes a that the IVI engine defines, a that the instrument class def attributes that are specific to particular instrument. Each c constant name begins with PREFIX_ATTR_, where PREI specific instrument prefix.
		For each IVI engine attribute specific driver include file us same constant name that ap ivi.h, except that the specific prefix replaces the IVI prefix. example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines th constant name:
		#define FL45_ATTR_CACH IVI_ATTR_CACHE
		For each instrument class at specific driver include file us same constant name that ap instrument class include file, the specific instrument prefix

the class prefix. For example class include file, ividmm.h, (IVIDMM_ATTR_RANGE, an defines the following constar

#define FL45_ATTR_RANG IVIDMM_ATTR_RANGE

For each specific instrument the specific driver include file constant name and assigns is an offset from

IVI_SPECIFIC_PUBLIC_ATT For example, fl45.h defines t constant name:

#define

FL45_ATTR_HOLD_THRE: (IVI_SPECIFIC_PUBLIC_A' + 3L)

For each attribute that is privinstrument driver, the instrumisource file defines a constar assigns a value that is an off IVI_SPECIFIC_PRIVATE_AT^T For example, hp34401a.c defi following constant name:

#define

HP34401A_ATTR_TRIGGE (IVI_SPECIFIC_PRIVATE_1 + 1L)

readCallback ReadAttrViInt32_CallbackPtr want the IVI engine to invoke request the current value of attribute.

You must define the read cal function in the source code f specific instrument driver. Th must have the following prot

ViStatus _VI_FUNC Callback(vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViInt32 *value);

Upon entry to the callback, *va the cache value. Upon exit fron callback, *value must contain tl current value.

Note If you want to us IVI Specific Driver Attri dialog box to develop y instrument driver source retain the parameter n shown in the prototype callback.

If you do not want to use a recallback function, pass $\rm VI_N$

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrReadCallbackViInt64

Usage

ViStatus Ivi_SetAttrReadCallbackViInt64(ViSession vi, ViAttr attributeID, ReadAttrViInt64_CallbackPtr readCallback);

Purpose

This function sets the read callback function for a ViInt64 attribute. The IVI engine calls the read callback function when you request the current value of the attribute and the cache value is invalid.

If you do not want the IVI engine to invoke a read callback, specify VI_NULL for the Read Callback parameter.

You can set the read callback function when you create the attribute with Ivi_AddAttributeViInt64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that ye from Ivi_SpecificDriverNew. 7 identifies a particular IVI ses
attributeID	ViAttr	Pass the ID of an attribute fc parameter.
		The include file for a specific driver defines constant name the user-accessible attribute to the driver. This includes a that the IVI engine defines, a that the instrument class def attributes that are specific to particular instrument. Each c constant name begins with PREFIX_ATTR_, where PREI specific instrument prefix.
		For each IVI engine attribute specific driver include file us same constant name that ap ivi.h, except that the specific prefix replaces the IVI prefix. example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines th constant name:
		#define FL45_ATTR_CACH IVI_ATTR_CACHE
		For each instrument class at specific driver include file us same constant name that ap instrument class include file, the specific instrument prefix

the class prefix. For example class include file, ividmm.h, (IVIDMM_ATTR_RANGE, an defines the following constar

#define FL45_ATTR_RANG IVIDMM_ATTR_RANGE

For each specific instrument the specific driver include file constant name and assigns is an offset from

IVI_SPECIFIC_PUBLIC_ATT For example, fl45.h defines t constant name:

#define

FL45_ATTR_HOLD_THRE: (IVI_SPECIFIC_PUBLIC_A' + 3L)

For each attribute that is privinstrument driver, the instrumisource file defines a constar assigns a value that is an off IVI_SPECIFIC_PRIVATE_AT^T For example, hp34401a.c defi following constant name:

#define

HP34401A_ATTR_TRIGGE (IVI_SPECIFIC_PRIVATE_1 + 1L)

readCallback ReadAttrViInt64_CallbackPtr Specify the read callback fur want the IVI engine to invoke request the current value of attribute.

You must define the read cal function in the source code f specific instrument driver. Th must have the following prot

ViStatus _VI_FUNC Callback(vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViInt32 *value);

Upon entry to the callback, * contains the cache value. Up from the callback, *value mus the actual current value.

Note If you want to us IVI Specific Driver Attri dialog box to develop y instrument driver source retain the parameter n shown in the prototype callback.

If you do not want to use a recallback function, pass VI_N

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrReadCallbackViReal64

Usage

ViStatus Ivi_SetAttrReadCallbackViReal64(ViSession vi, ViAttr attributeID, ReadAttrViReal64_CallbackPtr readCallback);

Purpose

This function sets the read callback function for a ViReal64 attribute. The IVI engine calls the read callback function when you request the current value of the attribute and the cache value is invalid.

If you do not want the IVI engine to invoke a read callback, specify VI_NULL for the Read Callback parameter.

You can set the read callback function when you create the attribute with Ivi_AddAttributeViReal64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that from Ivi_SpecificDriverNew identifies a particular IVI se
attributeID	ViAttr	Pass the ID of an attribute parameter.
		The include file for a speci driver defines constant nat the user-accessible attribu- to the driver. This includes that the IVI engine defines that the instrument class d attributes that are specific particular instrument. Each constant name begins with PREFIX_ATTR_, where PR specific instrument prefix.
		For each IVI engine attribus specific driver include file us same constant name that ivi.h, except that the speci- prefix replaces the IVI prefix example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines constant name:
		#define FL45_ATTR_CAC IVI_ATTR_CACHE
		For each instrument class specific driver include file is same constant name that instrument class include fil the specific instrument pre
the class prefix. For exam class include file, ividmm.h IVIDMM_ATTR_RANGE, defines the following const

#define FL45_ATTR_RAN IVIDMM_ATTR_RANGE

For each specific instrume the specific driver include constant name and assign is an offset from IVI_SPECIFIC_PUBLIC_A^T For example, fl45.h define constant name:

#define

FL45_ATTR_HOLD_THR (IVI_SPECIFIC_PUBLIC_ + 3L)

For each attribute that is p instrument driver, the instr source file defines a const assigns a value that is an IVI_SPECIFIC_PRIVATE_A For example, hp34401a.c d following constant name:

#define
HP34401A_ATTR_TRIGC
(IVI_SPECIFIC_PRIVATE
+ 1L)

readCallback ReadAttrViReal64_CallbackPtr Specify the read callback 1 want the IVI engine to invorrequest the current value (attribute. You must define the read (function in the source code

> specific instrument driver. must have the following pr

ViStatus _VI_FUNC Callbac vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViReal64 *value);

Upon entry to the callback contains the cache value. from the callback, *value n the actual current value.

Note If you want to IVI Specific Driver A dialog box to develo instrument driver sou retain the parameter shown in the prototy callback.

If you do not want to use a callback function, pass VI_

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrReadCallbackViSession

Usage

ViStatus Ivi_SetAttrReadCallbackViSession(ViSession vi, ViAttr attributeID, ReadAttrViSession_CallbackPtr readCallback);

Purpose

This function sets the read callback function for a ViSession attribute. The IVI engine calls the read callback function when you request the current value of the attribute and the cache value is invalid.

If you do not want the IVI engine to invoke a read callback, specify VI_NULL for the Read Callback parameter.

You can set the read callback function when you create the attribute with Ivi_AddAttributeViSession.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tha from Ivi_SpecificDriverNev identifies a particular IVI s
attributeID	ViAttr	Pass the ID of an attribute parameter.
		The include file for a spec driver defines constant na the user-accessible attrib to the driver. This include that the IVI engine define that the instrument class attributes that are specific particular instrument. Eac constant name begins wil PREFIX_ATTR_, where P specific instrument prefix.
		For each IVI engine attrib specific driver include file same constant name that ivi.h, except that the spec prefix replaces the IVI pre example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include f the specific instrument pr

the class prefix. For exan class include file, ividmm. IVIDMM_ATTR_RANGE, defines the following cons

#define FL45_ATTR_RA IVIDMM_ATTR_RANG

For each specific instrum the specific driver include constant name and assig is an offset from IVI_SPECIFIC_PUBLIC_A For example, fl45.h define constant name:

#define

FL45_ATTR_HOLD_TH (IVI_SPECIFIC_PUBLIC + 3L)

For each attribute that is | instrument driver, the inst source file defines a cons assigns a value that is an IVI_SPECIFIC_PRIVATE_ For example, hp34401a.c (following constant name:

#define HP34401A_ATTR_TRIG (IVI_SPECIFIC_PRIVAT + 1L)

readCallback ReadAttrViSession_CallbackPtr Specify the read callback want the IVI engine to inv request the current value attribute. You must define the read

function in the source coc specific instrument driver. must have the following p

ViStatus _VI_FUNC Callba vi,ViSession io, ViConstString repCapName ViAttr attributeId, ViSession *value);

Upon entry to the callbacl contains the cache value. from the callback, *value the actual current value.

Note If you want to IVI Specific Driver A dialog box to develo instrument driver so retain the paramete shown in the protot callback.

If you do not want to use callback function, pass $\ensuremath{\mathrm{V}}\xspace$

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrReadCallbackViString

Usage

ViStatus Ivi_SetAttrReadCallbackViString(ViSession vi, ViAttr attributeID, ReadAttrViString_CallbackPtr readCallback);

Purpose

This function sets the read callback function for a ViString attribute. The IVI engine calls the read callback function when you request the current value of the attribute and the cache value is invalid.

If you do not want the IVI engine to invoke a read callback, specify VI_NULL for the Read Callback parameter.

You can set the read callback function when you create the attribute with Ivi_AddAttributeViString.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that y from Ivi_SpecificDriverNew. identifies a particular IVI set
attributeID	ViAttr	Pass the ID of an attribute f parameter.
		The include file for a specifi driver defines constant nam the user-accessible attribute to the driver. This includes a that the IVI engine defines, that the instrument class de attributes that are specific to particular instrument. Each constant name begins with PREFIX_ATTR_, where PRE specific instrument prefix.
		For each IVI engine attribut specific driver include file us same constant name that a ivi.h, except that the specific prefix replaces the IVI prefix example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines the constant name:
		#define FL45_ATTR_CACH IVI_ATTR_CACHE
		For each instrument class a specific driver include file us same constant name that a instrument class include file the specific instrument prefi

the class prefix. For exampl class include file, ividmm.h, IVIDMM_ATTR_RANGE, ai defines the following consta

#define FL45_ATTR_RAN(IVIDMM_ATTR_RANGE

For each specific instrumen the specific driver include fil constant name and assigns is an offset from IVI_SPECIFIC_PUBLIC_AT^{*}

For example, fl45.h defines constant name:

#define

FL45_ATTR_HOLD_THRE (IVI_SPECIFIC_PUBLIC_/ + 3L)

For each attribute that is pri instrument driver, the instru source file defines a consta assigns a value that is an o IVI_SPECIFIC_PRIVATE_AI For example, hp34401a.c de following constant name:

#define

HP34401A_ATTR_TRIGGI (IVI_SPECIFIC_PRIVATE_ + 1L)

readCallback ReadAttrViString_CallbackPtr Specify the read callback fu want the IVI engine to invok request the current value of attribute.

You must define the read ca function in the source code specific instrument driver. T must have the following pro

ViStatus _VI_FUNC Callback vi,ViSession io, ViConstString repCapName,

ViAttr attributeId,

const ViConstString cacheValu

Unlike the read callback fun the other data types, you dc the current value to the call the last parameter. Instead, the current value by passin Ivi_SetValInStringCallback in function.

Note If you want to u IVI Specific Driver Att dialog box to develop instrument driver sour retain the parameter r shown in the prototyp callback.

If you do not want to use a I callback function, pass VI_I

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrWriteCallbackViAddr

Usage

ViStatus Ivi_SetAttrWriteCallbackViAddr(ViSession vi, ViAttr attributeID, WriteAttrViAddr_CallbackPtr writeCallback);

Purpose

This function sets the write callback function for a ViAddr attribute. The IVI engine calls the write callback function when you specify a new value for the attribute and the cache value is invalid or is not equal to the new value.

If you do not want the IVI engine to invoke a write callback, specify VI_NULL for the Write Callback parameter.

You can set the write callback function when you create the attribute with Ivi_AddAttributeViAddr.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that y from Ivi_SpecificDriverNew. identifies a particular IVI ses
attributeID	ViAttr	Pass the ID of an attribute fc parameter.
		The include file for a specific driver defines constant name the user-accessible attribute to the driver. This includes a that the IVI engine defines, a that the instrument class def attributes that are specific to particular instrument. Each c constant name begins with PREFIX_ATTR_, where PREI specific instrument prefix.
		For each IVI engine attribute specific driver include file us same constant name that ap ivi.h, except that the specific prefix replaces the IVI prefix example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines th constant name:
		#define FL45_ATTR_CACH IVI_ATTR_CACHE
		For each instrument class at specific driver include file us same constant name that ap instrument class include file, the specific instrument prefix

the class prefix. For example class include file, ividmm.h, (IVIDMM_ATTR_RANGE, an defines the following constar

#define FL45_ATTR_RANG IVIDMM_ATTR_RANGE

For each specific instrument the specific driver include file constant name and assigns is an offset from

IVI_SPECIFIC_PUBLIC_ATT For example, fl45.h defines t constant name:

#define

FL45_ATTR_HOLD_THRE: (IVI_SPECIFIC_PUBLIC_A + 3L)

For each attribute that is privinstrument driver, the instrumination source file defines a constar assigns a value that is an off IVI_SPECIFIC_PRIVATE_AT For example, hp34401a.c def following constant name:

#define

HP34401A_ATTR_TRIGGE (IVI_SPECIFIC_PRIVATE_4 + 1L)

writeCallback WriteAttrViAddr_CallbackPtr Specify the write callback fur want the IVI engine to invoke set the attribute to a new val

You must define the write ca function in the source code f specific instrument driver. Th must have the following prot

ViStatus _VI_FUNC Callback(vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViAddr value);



Note If you want to u IVI Specific Driver Attr dialog box to develop y instrument driver source retain the parameter n shown in the prototype callback.

If you do not want to use a w callback function, pass VI_N

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrWriteCallbackViBoolean

Usage

ViStatus Ivi_SetAttrWriteCallbackViBoolean(ViSession vi, ViAttr attributeID, WriteAttrViBoolean_CallbackPtr writeCallback);

Purpose

This function sets the write callback function for a ViBoolean attribute. The IVI engine calls the write callback function when you specify a new value for the attribute and the cache value is invalid or is not equal to the new value.

If you do not want the IVI engine to invoke a write callback, specify VI_NULL for the Write Callback parameter.

You can set the write callback function when you create the attribute with lvi_AddAttributeViBoolean.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle the from Ivi_SpecificDriverNe identifies a particular IVI
attributeID	ViAttr	Pass the ID of an attribut parameter.
		The include file for a spe driver defines constant n the user-accessible attrik to the driver. This include that the IVI engine define that the instrument class attributes that are specifi particular instrument. Ea constant name begins wi PREFIX_ATTR_, where F specific instrument prefix
		For each IVI engine attrik specific driver include file same constant name tha ivi.h, except that the spec prefix replaces the IVI pre example, ivi.h defines IVI_ATTR_CACHE, and 1 include file, f145.h, define constant name:
		#define FL45_ATTR_CA IVI_ATTR_CACHE
		For each instrument clas specific driver include file same constant name tha instrument class include the specific instrument p

the class prefix. For exar class include file, ividmm IVIDMM_ATTR_RANGE defines the following con

#define FL45_ATTR_RA IVIDMM_ATTR_RANG

For each specific instrum the specific driver include constant name and assig is an offset from IVI_SPECIFIC_PUBLIC_1

For example, fl45.h defin constant name:

#define

FL45_ATTR_HOLD_TH (IVI_SPECIFIC_PUBLI(+ 3L)

For each attribute that is instrument driver, the ins source file defines a con: assigns a value that is ar IVI_SPECIFIC_PRIVATE_ For example, hp34401a.c following constant name:

#define

HP34401A_ATTR_TRIC (IVI_SPECIFIC_PRIVAT + 1L)

writeCallback WriteAttrViBoolean_CallbackPtr Specify the write callback want the IVI engine to inviset the attribute to a new

You must define the write function in the source cospecific instrument driver must have the following p

ViStatus _VI_FUNC Callba vi,ViSession io, ViConstString repCapNama ViAttr attributeId, ViBoolean value);

Note If you want to IVI Specific Driver dialog box to devel instrument driver so retain the parameter shown in the protot callback.

If you do not want to use callback function, pass $\ensuremath{\mathbb{V}}$

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrWriteCallbackViInt32

Usage

ViStatus Ivi_SetAttrWriteCallbackViInt32(ViSession vi, ViAttr attributeID, WriteAttrViInt32_CallbackPtr writeCallback);

Purpose

This function sets the write callback function for a ViInt32 attribute. The IVI engine calls the write callback function when you specify a new value for the attribute and the cache value is invalid or is not equal to the new value.

If you do not want the IVI engine to invoke a write callback, specify VI_NULL for the Write Callback parameter.

You can set the write callback function when you create the attribute with Ivi_AddAttributeViInt32.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that y from Ivi_SpecificDriverNew.
attributeID	ViAttr	Pass the ID of an attribute for parameter.
		The include file for a specific driver defines constant name the user-accessible attribute to the driver. This includes a that the IVI engine defines, a that the instrument class def attributes that are specific to particular instrument. Each o constant name begins with PREFIX_ATTR_, where PRE specific instrument prefix.
		For each IVI engine attribute specific driver include file us same constant name that ap ivi.h, except that the specific prefix replaces the IVI prefix example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines th constant name:
		#define FL45_ATTR_CACH IVI_ATTR_CACHE
		For each instrument class at specific driver include file us same constant name that ar instrument class include file, the specific instrument prefix

the class prefix. For example class include file, ividmm.h, (IVIDMM_ATTR_RANGE, an defines the following constal

#define FL45_ATTR_RANC IVIDMM_ATTR_RANGE

For each specific instrument the specific driver include file constant name and assigns is an offset from

IVI_SPECIFIC_PUBLIC_ATT For example, fl45.h defines t constant name:

#define
FL45_ATTR_HOLD_THRE
(IVI_SPECIFIC_PUBLIC_A
+ 3L)

For each attribute that is privinstrument driver, the instrur source file defines a constar assigns a value that is an of IVI_SPECIFIC_PRIVATE_AT For example, hp34401a.c def following constant name:

#define

HP34401A_ATTR_TRIGGE (IVI_SPECIFIC_PRIVATE_ + 1L)

writeCallback WriteAttrViInt32_CallbackPtr Specify the write callback fu want the IVI engine to invok set the attribute to a new val

> You must define the write ca function in the source code 1 specific instrument driver. Th must have the following prot

ViStatus _VI_FUNC Callback(vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViInt32 value);



Note If you want to us IVI Specific Driver Attr dialog box to develop instrument driver sourretain the parameter n shown in the prototype callback.

If you do not want to use a v callback function, pass VI_N

Return Value

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Ivi_SetAttrWriteCallbackViInt64

Usage

ViStatus Ivi_SetAttrWriteCallbackViInt64(ViSession vi, ViAttr attributeID, WriteAttrViInt64_CallbackPtr writeCallback);

Purpose

This function sets the write callback function for a VIInt64 attribute. The IVI engine calls the write callback function when you specify a new value for the attribute and the cache value is invalid or is not equal to the new value.

If you do not want the IVI engine to invoke a write callback, specify VI_NULL for the Write Callback parameter.

You can set the write callback function when you create the attribute with Ivi_AddAttributeViInt64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that y from Ivi_SpecificDriverNew.
attributeID	ViAttr	Pass the ID of an attribute for parameter.
		The include file for a specific driver defines constant name the user-accessible attribute to the driver. This includes a that the IVI engine defines, a that the instrument class def attributes that are specific to particular instrument. Each o constant name begins with PREFIX_ATTR_, where PRE specific instrument prefix.
		For each IVI engine attribute specific driver include file us same constant name that ap ivi.h, except that the specific prefix replaces the IVI prefix example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines th constant name:
		#define FL45_ATTR_CACH IVI_ATTR_CACHE
		For each instrument class at specific driver include file us same constant name that ar instrument class include file, the specific instrument prefix
the class prefix. For example class include file, ividmm.h, (IVIDMM_ATTR_RANGE, an defines the following constal

#define FL45_ATTR_RANC IVIDMM_ATTR_RANGE

For each specific instrument the specific driver include file constant name and assigns is an offset from

IVI_SPECIFIC_PUBLIC_ATT For example, fl45.h defines t constant name:

#define
FL45_ATTR_HOLD_THRE
(IVI_SPECIFIC_PUBLIC_A
+ 3L)

For each attribute that is privinstrument driver, the instrur source file defines a constar assigns a value that is an of IVI_SPECIFIC_PRIVATE_AT For example, hp34401a.c def following constant name:

#define

HP34401A_ATTR_TRIGGE (IVI_SPECIFIC_PRIVATE_ + 1L)

writeCallback WriteAttrViInt64_CallbackPtr Specify the write callback fu want the IVI engine to invok set the attribute to a new val

You must define the write ca function in the source code 1 specific instrument driver. Th must have the following prot

ViStatus _VI_FUNC Callback(vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViInt32 value);



Note If you want to us IVI Specific Driver Attr dialog box to develop instrument driver sourretain the parameter n shown in the prototype callback.

If you do not want to use a v callback function, pass VI_N

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetAttrWriteCallbackViReal64

Usage

ViStatus Ivi_SetAttrWriteCallbackViReal64(ViSession vi, ViAttr attributeID, WriteAttrViReal64_CallbackPtr writeCallback);

Purpose

This function sets the write callback function for a ViReal64 attribute. The IVI engine calls the write callback function when you specify a new value for the attribute and the cache value is invalid or is not equal to the new value.

If you do not want the IVI engine to invoke a write callback, specify VI_NULL for the Write Callback parameter.

You can set the write callback function when you create the attribute with Ivi_AddAttributeViReal64.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle tha from Ivi_SpecificDriverNew identifies a particular IVI s
attributeID	ViAttr	Pass the ID of an attribute parameter.
		The include file for a spec driver defines constant na the user-accessible attribu to the driver. This includes that the IVI engine defines that the instrument class c attributes that are specific particular instrument. Each constant name begins with PREFIX_ATTR_, where PF specific instrument prefix.
		For each IVI engine attribution specific driver include file same constant name that ivi.h, except that the speci- prefix replaces the IVI pre- example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA(IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include fi the specific instrument pre

the class prefix. For exam class include file, ividmm.l IVIDMM_ATTR_RANGE, defines the following cons

#define FL45_ATTR_RANCE

For each specific instrume the specific driver include constant name and assigr is an offset from IVI_SPECIFIC_PUBLIC_A'

For example, fl45.h define constant name:

#define

FL45_ATTR_HOLD_THF (IVI_SPECIFIC_PUBLIC_ + 3L)

For each attribute that is p instrument driver, the instr source file defines a const assigns a value that is an IVI_SPECIFIC_PRIVATE_/ For example, hp34401a.c c following constant name:

#define

HP34401A_ATTR_TRIG((IVI_SPECIFIC_PRIVATI + 1L)

writeCallback WriteAttrViReal64_CallbackPtr Specify the write callback want the IVI engine to inverset the attribute to a new You must define the write

function in the source cod specific instrument driver. must have the following p

ViStatus _VI_FUNC Callbac vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViReal64 value);



Note If you want to IVI Specific Driver A dialog box to develo instrument driver so retain the parameter shown in the prototy callback.

If you do not want to use a callback function, pass VI

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetAttrWriteCallbackViSession

Usage

ViStatus Ivi_SetAttrWriteCallbackViSession(ViSession vi, ViAttr attributeID, WriteAttrViSession_CallbackPtr writeCallback);

Purpose

This function sets the write callback function for a ViSession attribute. The IVI engine calls the write callback function when you specify a new value for the attribute and the cache value is invalid or is not equal to the new value.

If you do not want the IVI engine to invoke a write callback, specify VI_NULL for the Write Callback parameter.

You can set the write callback function when you create the attribute with Ivi_AddAttributeViSession.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle the from Ivi_SpecificDriverNe identifies a particular IVI
attributeID	ViAttr	Pass the ID of an attribut parameter.
		The include file for a speedriver defines constant nathe user-accessible attributes to the driver. This include that the IVI engine define that the IVI engine define that the instrument class attributes that are specific particular instrument. Eac constant name begins wi PREFIX_ATTR_, where P specific instrument prefix
		For each IVI engine attrik specific driver include file same constant name that ivi.h, except that the spec prefix replaces the IVI pre example, ivi.h defines IVI_ATTR_CACHE, and t include file, fl45.h, defines constant name:
		#define FL45_ATTR_CA IVI_ATTR_CACHE
		For each instrument class specific driver include file same constant name that instrument class include the specific instrument pr

the class prefix. For exan class include file, ividmm IVIDMM_ATTR_RANGE, defines the following con

#define FL45_ATTR_RA IVIDMM_ATTR_RANG

For each specific instrum the specific driver include constant name and assig is an offset from IVI_SPECIFIC_PUBLIC_# For example, fl45.h define constant name:

#define
FL45_ATTR_HOLD_TH
(IVI_SPECIFIC_PUBLI(
+ 3L)

For each attribute that is instrument driver, the inst source file defines a cons assigns a value that is ar IVI_SPECIFIC_PRIVATE_ For example, hp34401a.c following constant name:

#define

HP34401A_ATTR_TRIG (IVI_SPECIFIC_PRIVAT + 1L)

writeCallback WriteAttrViSession_CallbackPtr Specify the write callback want the IVI engine to inviset the attribute to a new You must define the write

function in the source co specific instrument driver must have the following p

ViStatus _VI_FUNC Callba vi,ViSession io, ViConstString repCapName ViAttr attributeId, ViSession value);



Note If you want to IVI Specific Driver *i* dialog box to develinstrument driver so retain the paramete shown in the protot callback.

If you do not want to use callback function, pass $\rm V$

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetAttrWriteCallbackViString

Usage

ViStatus Ivi_SetAttrWriteCallbackViString(ViSession vi, ViAttr attributeID, WriteAttrViString_CallbackPtr writeCallback);

Purpose

This function sets the write callback function for a ViString attribute. The IVI engine calls the write callback function when you specify a new value for the attribute and the cache value is invalid or is not equal to the new value.

If you do not want the IVI engine to invoke a write callback, specify VI_NULL for the Write Callback parameter.

You can set the write callback function when you create the attribute with Ivi_AddAttributeViString.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that from Ivi_SpecificDriverNew. identifies a particular IVI se
attributeID	ViAttr	Pass the ID of an attribute the parameter.
		The include file for a specific driver defines constant name the user-accessible attribute to the driver. This includes a that the IVI engine defines, that the IVI engine defines, that the instrument class defines attributes that are specific to particular instrument. Each constant name begins with PREFIX_ATTR_, where PRI specific instrument prefix.
		For each IVI engine attribut specific driver include file u same constant name that a ivi.h, except that the specifi prefix replaces the IVI prefit example, ivi.h defines IVI_ATTR_CACHE, and the include file, fl45.h, defines t constant name:
		#define FL45_ATTR_CAC1 IVI_ATTR_CACHE
		For each instrument class a specific driver include file u same constant name that a instrument class include file the specific instrument pref

the class prefix. For examp class include file, ividmm.h, IVIDMM_ATTR_RANGE, a defines the following consta

#define FL45_ATTR_RAN(IVIDMM_ATTR_RANGE

For each specific instrumer the specific driver include fi constant name and assigns is an offset from

IVI_SPECIFIC_PUBLIC_AT For example, fl45.h defines constant name:

#define

FL45_ATTR_HOLD_THRI (IVI_SPECIFIC_PUBLIC_. + 3L)

For each attribute that is pr instrument driver, the instru source file defines a consta assigns a value that is an o IVI_SPECIFIC_PRIVATE_A^T For example, hp34401a.c de following constant name:

#define

HP34401A_ATTR_TRIGG (IVI_SPECIFIC_PRIVATE_ + 1L)

writeCallback WriteAttrViString_CallbackPtr Specify the write callback fi want the IVI engine to invol set the attribute to a new va

> You must define the write c function in the source code specific instrument driver. T must have the following prc

ViStatus _VI_FUNC Callback vi,ViSession io, ViConstString repCapName, ViAttr attributeId, ViConstString value);

Note If you want to u IVI Specific Driver Att dialog box to develop instrument driver sou retain the parameter shown in the prototyp callback.

If you do not want to use a callback function, pass VI_

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetErrorInfo

Usage

ViStatus Ivi_SetErrorInfo(ViSession vi, ViBoolean overwrite, ViStatus primaryError, ViStatus secondaryError, ViChar elaboration[]);

Purpose

This function sets the error information for the current execution thread and the IVI session you specify. If you pass VI_NULL for the vi parameter, this function sets the error information only for the current execution thread.

The error information includes a primary error code, secondary error code, and an error elaboration string. For a particular session, this information is the same as the values held in the following attributes:

- IVI_ATTR_PRIMARY_ERROR or PREFIX_ATTR_PRIMARY_ERROR
- IVI_ATTR_SECONDARY_ERROR or PREFIX_ATTR_SECONDARY_ERROR
- IVI_ATTR_ERROR_ELABORATION or PREFIX_ATTR_ERROR_ELABORATION

The IVI engine also maintains this error information separately for each thread. This is useful if you do not have a session handle to pass to Ivi_SetErrorInfo or Ivi_GetErrorInfo, which occurs when a call to Ivi_SpecificDriverNew fails.

The IVI engine retains the information you specify with Ivi_SetErrorInfo until the end-user retrieves it by calling PREFIX_GetError, the end-user clears it by calling PREFIX_ClearError, or you overwrite it with another call to Ivi_SetErrorInfo. Ivi_GetErrorInfo, which the end-user calls through PREFIX_GetError, always clears the error information.

Normally, it is the responsibility of the end-user to decide when to clear the error information by calling PREFIX_GetError or PREFIX_ClearError. If an instrument driver calls Ivi_GetErrorInfo, it should restore the error information by calling Ivi_SetErrorInfo, possibly adding a secondary error code or elaboration string.

Ivi_SetErrorInfo does not overwrite existing significant error information unless you pass VI_TRUE for the Overwrite parameter. Typically, you pass VI_FALSE for this parameter so you can make multiple calls to this function at different levels in your instrument driver source code without the risk of losing important error information. Also, end-users can make multiple calls to the instrument driver and be assured that PREFIX_GetError returns significant information about the first error that occurred since their last call to PREFIX_GetError or PREFIX_ClearError.

The viCheckErr, viCheckErrElab, viCheckParm, viCheckAlloc, and viCheckWarn macros use Ivi_SetErrorInfo. The ivi.h include file defines these macros. The viCheckWarn macro calls Ivi_SetErrorInfo on both warnings and errors, whereas the other macros discard warnings and call Ivi_SetErrorInfo only on errors.

Parameters		
Name	Туре	Description
vi	ViSession	To set the error information for a particular IVI session, pass the ViSession handle that you obtain from Ivi_SpecificDriverNew. When you pass a ViSession handle, the function also sets the error information for the current thread.
		To set only the error information for the current thread, pass VI_NULL.
overwrite	ViBoolean	Pass VI_TRUE (1) for this parameter if you want the new error information to overwrite the existing error information regardless of the current contents of the error information.
		If you pass VI_FALSE (0), the function uses the following logic to determine whether to overwrite the existing error information.
		 It overwrites the primary error code if either:
		 the existing primary code is VI_SUCCESS
		 the existing primary code is a positive warning code and the primary error code you specify is a negative error code.
		It overwrites the secondary error code if either:
		 it overwrites the old primary error code with a different value
		 the existing secondary code is VI_SUCCESS and the

primary code you specify is either VI_SUCCESS or equal to the old primary error code.

- 3. It overwrites the elaboration string if either:
 - it overwrites the old primary error code with a different value
 - the existing elaboration string is empty and the primary code you specify is either VI_SUCCESS or equal to the old primary error code.

This behavior allows you to make multiple calls to Ivi_SetErrorInfo at different levels in your instrument driver source code without the risk of losing important error information. For instance, if you have already set the primary code to a negative error value, subsequent calls to Ivi_SetErrorInfo do not change the value. Consequently, Ivi_GetErrorInfo always returns the first error that you reported.

At the same time, you can make subsequent calls to Ivi_SetErrorInfo to add further information. If your first call to Ivi_SetErrorInfo specifies a negative primary error code, a zero secondary error code, and no elaboration string, you can later add a secondary error code and an elaboration string by calling Ivi_SetErrorInfo with the same primary error code.

Normally, end-users expect the error information to describe the first error that occurred since their last call to PREFIX_GetError or PREFIX_ClearError. So avoid passing VI_TRUE for this parameter.

- primaryErrorViStatusSpecify a status code describing the
primary error condition. Use VI_SUCCESS
(0) to indicate no error or warning. Use a
positive value to indicate a warning. Use a
negative value to indicate an error.secondaryErrorViStatusSpecify a status code that further describes
the error or warning condition. If you have
no further description, pass VI_SUCCESS
(0) for this parameter.elaborationViChar[]Specify an olaboration string that further
- **elaboration** VICnar[] Specify an elaboration string that further describes the error or warning condition. The IVI engine stores the entire string for the session you specify, but it retains only IVI_MAX_MESSAGE_BUF_SIZE-1 (255) characters for the current execution thread.

If you have no further description, pass VI_NULL or an empty string.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetNeedToCheckStatus

Usage

ViStatus Ivi_SetNeedToCheckStatus(ViSession vi, ViBoolean needToCheckStatus);

Purpose

This function allows an instrument driver to indicate whether it is necessary to check the status of the instrument.

The IVI engine maintains an internal needToCheckStatus variable for each session indicating whether it is necessary to check the status of the instrument. When you create a new session, the initial value of the variable is VI_TRUE. The IVI engine sets the needToCheckStatus variable to VI_TRUE when it invokes the read callback or write callback for an attribute for which the IVI_VAL_DONT_CHECK_STATUS flag is 0. The Ivi_WriteInstrData and Ivi_WriteFromFile functions also set the variable to VI_TRUE. The IVI engine sets the variable to VI_TRUE. The IVI engine sets the variable to VI_TRUE with the IVI_VAL_DONT_CHECK_STATUS flag is 0. The Ivi_WriteInstrData and Ivi_WriteFromFile functions also set the variable to VI_TRUE. The IVI engine sets the variable to VI_FALSE after it invokes the check status callback successfully.

The Ivi_SetNeedToCheckStatus function allows an instrument driver to set the state of the internal needToCheckStatus variable. A driver typically sets the variable to VI_TRUE before it attempts direct instrument I/O. It sets it to VI_FALSE after it calls the check status callback successfully.



Note Do not call this function unless you have already locked the session.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
needToCheckStatus	ViBoolean	Pass VI_TRUE before you attempt to interact with the instrument directly rather than through Ivi_SetAttribute, Ivi_GetAttribute, Ivi_WriteInstrData or Ivi_WriteFromFile functions.
		Pass VI_FALSE after you invoke the check status callback successfully.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetRangeTableEnd

Usage

ViStatus Ivi_SetRangeTableEnd(IviRangeTablePtr rangeTable, ViInt32 index);

Purpose

This function sets the termination entry for a dynamic range table you create with Ivi_RangeTableNew. Ivi_RangeTableNew automatically sets the last entry you create to be the termination entry. For example, if you specify 10 entries, Ivi_RangeTableNew marks the entry at index 9 to be the termination entry. Use this function if you want to move the termination entry to a lower index.

Parameters

Name	Туре	Description
rangeTable	IviRangeTablePtr	Pass the range table pointer you obtain from Ivi_RangeTableNew.
index	Vilnt32	Specify the 0-based index of the entry to be the termination entry.
		Ivi_RangeTableNew automatically sets the last entry you create to be the termination entry. For example, if you specify 10 entries, Ivi_RangeTableNew marks the entry at index 9 to be the termination entry. Use this function if you want to move the termination entry to a lower index.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic
Ivi_SetRangeTableEntry

Usage

ViStatus Ivi_SetRangeTableEntry(IviRangeTablePtr rangeTable, ViInt32 index, ViReal64 discreteOrMinValue, ViReal64 maxValue, ViReal64 coercedValue, ViChar cmdString[], ViInt32 cmdValue);

This function configures the values in a range table entry. To set the terminating entry, call Ivi_SetRangeTableEnd.

Parameters		
Name	Туре	Description
rangeTable	IviRangeTablePtr	Pass the range table pointer you obtain from Ivi_RangeTableNew.
index	Vilnt32	Pass the 0-based index of the range table entry you want to configure. For example, if you specify 10 entries when you call Ivi_RangeTableNew, you can call Ivi_SetRangeTableEntry with indexes 0 through 8. The termination entry is at index 9, unless you place it at a lower index using Ivi_SetRangeTableEnd.
		If you call Ivi_SetRangeTableEnd to change the location of the termination entry, the index parameter to Ivi_SetRangeTableEntry must be less than the index of the termination entry.
discreteOrMinValue	ViReal64	Pass the value you want to assign to the discreteOrMinValue field of the entry. Refer to the help for the Type Of Table parameter to Ivi_RangeTableNew for a discussion of how this field is used under the different range table types.
maxValue	ViReal64	Pass the value you want to

		assign to the maxValue field of the entry. Refer to the help for the Type Of Table parameter to Ivi_RangeTableNew for a discussion of how this field is used under the different range table types. Discrete tables do not use this field.
coercedValue	ViReal64	Pass the value you want to assign to the coercedValue field of the entry. Refer to the help for the Type Of Table parameter to Ivi_RangeTableNew for a discussion of how this field is used under the different range table types. Discrete and ranged tables do not use this field.
cmdString	ViChar[]	Specify the string you want to assign to the cmdString field of the range table entry. The cmdString field is optional. You can use it to hold the command string that the write callback sends to the instrument when you set the attribute to the value or range of values that the range table entry defines. If you do not want to associate a command string with the range table entry, pass VI_NULL. If you want to dynamically allocate the command strings,
		call Ivi_Alloc. Pass the pointer you obtain from Ivi_Alloc for

		this parameter. If you call Ivi_RangeTableFree to deallocate the range table, you can request that it call Ivi_Free on each non-NULL command string in the table.
cmdValue	Vilnt32	Specify the integer value you want to assign to the cmdValue field of the range table entry. The cmdValue field is optional.
		For a register-based instrument, you can use the cmdValue field to store the register value that the write callback sends to the instrument when you set the attribute to the value or range of values that the range table entry defines.
		For a message-based instrument, you can use the cmdValue field to hold a value that the attribute write callback formats into an instrument command string. You can use the customInfo field of the IviRangeTable structure to store the format string for the instrument command.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetStoredRangeTablePtr

Usage

ViStatus Ivi_SetStoredRangeTablePtr(ViSession vi, ViAttr attributeID, IviRangeTablePtr rangeTable);

Sets the range table for an attribute. You can specify a range table when you call Ivi_AddAttributeViInt32 or Ivi_AddAttributeViReal64 to create the attribute. Use this function to replace the original range table with a different one.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID	ViAttr	Pass the ID of an attribute for this parameter.
		The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix.
		For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:
		#define FL45_ATTR_CACHE IVI_ATTR_CACHE
		For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines

IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define

FL45_ATTR_HOLD_THRESHOLD \ (IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define

HP34401A_ATTR_TRIGGER_TYPE \ (IVI_SPECIFIC_PRIVATE_ATTR_BASE + 1L)

rangeTableIviRangeTablePtrSets the range table for an attribute. You
can specify a range table when you call
Ivi_AddAttributeViInt32 or
Ivi_AddAttributeViReal64 to create the
attribute. Use this function to replace the
original range table with a different one.Specify the address the range table that
you want to use for the attribute. If you
do not want a range table, pass
VI NULL.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_SetValInStringCallback

Usage

ViStatus Ivi_SetValInStringCallback(ViSession vi, ViAttr attributeID, ViChar value[]);

This function sets the value of a ViString attribute in the context of the read or coerce callback function for it.

All read functions for ViString attributes must use this function to report the new value of the attribute. All coerce functions for ViString attributes must use this function to report the coerced value.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
attributeID	ViAttr	Pass the attribute ID that the ViString attribute read or coerce callback receives.
value	ViChar[]	The value that you want to report from the read or coerce callback.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_Simulating

Usage

ViBoolean = Ivi_Simulating(ViSession vi);

This function returns the current value of the IVI_ATTR_SIMULATE attribute for the session you specify.

Use Ivi_Simulating in the high-level functions in specific and class instrument drivers. Ivi_Simulating provides fast, convenient access to the IVI_ATTR_SIMULATE attribute because it does no error checking and does not lock the session.



Note Do not call this function unless you have already locked the session.

Parameters

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the IVI_ATTR_SIMULATE attribute for the session. This attribute determines whether or not to simulate instrument driver I/O operations.

Values:

VI_TRUE (1) Simulate

VI_FALSE (0) Do not simulate

If you pass an invalid session handle to the function, this parameter returns $\rm VI_FALSE.$

Related Topic

Ivi_SpecificDriverNew

Usage

ViStatus Ivi_SpecificDriverNew(ViChar specificDriverPrefix[], ViChar optionString[], ViSession vi);

This function performs the following initialization actions:

• Creates a new IVI session to a specific instrument driver and optionally sets the initial state of the following session attributes:

IVI_ATTR_RANGE_CHECKIVI_ATTR_QUERY_INSTRUMENT_STA IVI_ATTR_CACHE IVI_ATTR_SIMULATE IVI_ATTR_RECORD_COERCIONS IVI_ATTR_DRIVER_SETUP IVI_ATTR_INTERCHANGE_CHECK

• Returns a ViSession handle which you use in subsequent function calls to identify the session.

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 Ivi_LockSession and Ivi_UnlockSession functions to protect sections of code that require exclusive access to the resource.

Note This function does not create a VISA session to any instrument resources. If you use VISA to communicate to the instrument, you must create a VISA session yourself and set the IVI_ATTR_IO_SESSION attribute to that value. Otherwise, you can use the IVI_ATTR_IO_SESSION attribute to hold a handle to whatever communications resource you use.

Parameters

Name	Туре	Description	
specificDriverPrefix	ViChar[]	This parameter sp user-callable funct function name. Fo function named "fl	ecifies the prefix of the ion in the driver must r example, if the Fluke 45_init", then "fl45" is
optionString	nString ViChar[]		arameter to set the in le following table lists me you use in this pa
		Name	Attribute Defined C
		RangeCheck	IVI_ATTR_RANGE_C
		QueryInstrStatus	IVI_ATTR_QUERY_I
		Cache	IVI_ATTR_CACHE
		Simulate	IVI_ATTR_SIMULAT
		RecordCoercions	IVI_ATTR_RECORD_
		DriverSetup	IVI_ATTR_DRIVER_
		Interchange Check	IVI_ATTR_INTERCH
		If you pass NULL of the default values. value explicitly in a	or an empty string for You can override the a string you pass for th
		The format of an a column in the table	ssignment is, "Name= e above, and Value is
		To set the attribu To set the attribu	te to VI_TRUE, use V te to VI_FALSE, use י
		The function interp	prets the Name and Va
		To set multiple attr	ibutes, separate the a

You do not have to specify all of the attri the attributes, the session uses its defau



Note Normally, you use this funct

	and PREFIX_InitWithOptions in the PREFIX_init, you pass in an empty PREFIX_InitWithOptions, you pass that the user passed to PREFIX_Ir
vi ViS	Session Returns a ViSession handle that you use subsequent function calls.
	This function creates a new session eac if you have multiple physical instances c
	Avoid creating multiple concurrent sessi instrument. Although you can create mo same resource, it is best not to do so. A session in multiple execution threads. Yo Ivi_LockSession and Ivi_UnlockSession to require exclusive access to the resource

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_Spying

Usage

ViBoolean = Ivi_Spying(ViSession vi);

This function returns the current value of the IVI_ATTR_SPY attribute for the session you specify.

Use Ivi_Spying in the high-level functions in class instrument drivers. Ivi_Spying provides fast, convenient access to the IVI_ATTR_SPY attribute because it does no error checking and does not lock the session.



Note Do not call this function unless you have already locked the session.

Parameters

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the IVI_ATTR_SPY attribute for the session. This attribute determines whether class instrument drivers use the NI-Spy utility to record calls to class driver functions.

Values:

VI_TRUE (1) Spy on

VI_FALSE (0) Spy off

If you pass an invalid session handle to the function, this parameter returns $\rm VI_FALSE.$

Related Topic

Ivi_UnlockSession

Usage

ViStatus Ivi_UnlockSession(ViSession vi, ViBoolean* callerHasLock);

This function releases a lock that you acquired on an instrument session using Ivi_LockSession. Refer to Ivi_LockSession for additional information on session locks.

Instrument drivers export this function to the end-user through the PREFIX_UnlockSession function.

Parameters

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
callerHasLock	ViBoolean*	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. Initialize the local variable to VI_FALSE when you declare it. Pass the same address to any other calls you make to Ivi_LockSession or Ivi_UnlockSession in the same function.
		The parameter is an input/output parameter. Ivi_LockSession and Ivi_UnlockSession each inspect the current value and take the following actions:
		If the value is VI_TRUE, Ivi_LockSession does not lock the session again. If the value is VI_FALSE, Ivi_LockSession obtains the lock and sets the value of the parameter to VI_TRUE. If the value is VI_FALSE, Ivi_UnlockSession does not attempt to unlock the session. If the value is VI_TRUE, Ivi_UnlockSession unlocks the lock and sets the value of the parameter to VI_FALSE.

Thus, you can, call Ivi_UnlockSession at the

end of your function without worrying about whether you actually have the lock.

Example:

ViStatus PREFIX_Func (ViSession vi, ViInt32 flags){

```
ViStatus error = VI_SUCCESS;
ViBoolean haveLock = VI_FALSE;
```

if (flags & BIT_1)

```
{
viCheckErr( Ivi_LockSession(vi,
&haveLock));
viCheckErr( TakeAction1(vi));
```

if (flags & BIT_2)

```
{
viCheckErr( Ivi_UnlockSession(vi,
&haveLock));
viCheckErr( TakeAction2(vi));
viCheckErr( Ivi_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.

```
*/
```

Ivi_UnlockSession(vi, &haveLock);
return error;

}

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_UseSpecificSimulation

Usage

ViBoolean = Ivi_UseSpecificSimulation(ViSession vi);

This function returns the current value of the IVI_ATTR_USE_SPECIFIC_SIMULATION attribute for the session you specify.

Use Ivi_UseSpecificSimulation in the high-level functions in specific and class instrument drivers. Ivi_UseSpecificSimulation provides fast, convenient access to the IVI_ATTR_USE_SPECIFIC_SIMULATION attribute because it does no error checking and does not lock the session.



Note Do not call this function unless you have already locked the session.
Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Returns the value of the IVI_ATTR_SIMULATE_IN_SPECIFIC attribute for the session. This attribute controls whether the specific driver or the class driver simulates I/O operations when simulation is enabled.

Values:

VI_TRUE (1) Simulate in specific driver

VI_FALSE (0) Simulate in class driver

If you pass an invalid session handle to the function, this parameter returns $\rm VI_FALSE.$

Related Topic

Ivi_ValidateAttrForChannel

Usage

ViStatus Ivi_ValidateAttrForChannel(ViSession vi, ViChar channelName[], ViAttr attributeID);

Purpose

This function checks whether you can use an attribute on a particular channel. If either the AttributeID or Channel Name parameter is invalid for the session, Ivi_ValidateAttrForChannel returns an error. Otherwise, it checks for the following cases in which the combination of the AttributeID and Channel Name is invalid:

The channel name is VI_NULL or the empty string and the attribute is channel-based. An attribute is channel-based if its

IVI_VAL_MULTI_CHANNEL flag is set. In this case, the function returns the IVI_ERROR_REPEATED_CAPABILITY_NAME_REQUIRED error code.

The channel name refers to a specific channel and the attribute is not channel-based. In this case, the function returns the IVI_ERROR_REPEATED_CAPABILITY_NAME_NOT_ALLOWED error code.

The channel name refers to a specific channel, the attribute is channelbased, but the instrument driver calls Ivi_RestrictAttrToChannels to exclude the channel from using the attribute. In this case, the function returns the

IVI_ERROR_ATTR_NOT_VALID_FOR_REPEATED_CAPABILITY error code.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
channelName	ViChar[]	 Pass the channel name that you want to verify as valid for a particular channel. You can pass one of the following types of values: VI_NULL or an emptry string A specific driver channel string, which is one that the specific instrument driver specifies as valid using Ivi_BuildChannelTable or Ivi_AddToChannelTable. A virtual channel name that the end-user specifies in the IVI configuration file.
attributeID	ViAttr	Pass the ID of an attribute for this parameter. The include file for a specific instrument driver defines constant names for all of the user-accessible attributes that apply to the driver. This includes attributes that the IVI engine defines, attributes that the instrument class defines, and attributes that are specific to the particular instrument. Each defined constant name begins with PREFIX_ATTR_, where PREFIX is the specific instrument prefix. For each IVI engine attribute, the specific driver include file uses the same constant name that appears in ivi.h, except that the specific instrument prefix replaces the IVI

prefix. For example, ivi.h defines IVI_ATTR_CACHE, and the Fluke 45 include file, fl45.h, defines the following constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attribute, the specific driver include file uses the same constant name that appears in the instrument class include file, except that the specific instrument prefix replaces the class prefix. For example, the DMM class include file, ividmm.h, defines IVIDMM_ATTR_RANGE, and fl45.h defines the following constant name:

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument attribute, the specific driver include file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PUBLIC_ATTR_BASE. For example, fl45.h defines the following constant name:

#define FL45_ATTR_HOLD_THRESHOLD \
(IVI_SPECIFIC_PUBLIC_ATTR_BASE + 3L)

For each attribute that is private to an instrument driver, the instrument driver source file defines a constant name and assigns a value that is an offset from IVI_SPECIFIC_PRIVATE_ATTR_BASE. For example, hp34401a.c defines the following constant name:

#define HP34401A_ATTR_TRIGGER_TYPE \
(IVI_SPECIFIC_PRIVATE_ATTR_BASE +
1L)

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_ValidateAttrForRepCapName

Usage

ViStatus Ivi_ValidateAttrForRepCapName(ViSession vi, ViChar RepeatedCapabilityIdentifier[], ViAttr attributeID);

Purpose

This function checks whether you can use an attribute on a particular repeated capability instance. If either the AttributeID or Repeated Capability Name parameter is invalid for the session, this function returns an error. Otherwise, it checks for the following cases in which the combination of the AttributeID and Repeated Capability Name is invalid:

- The repeated capability name is VI_NULL or the empty string and the attribute applies to a repeated capability. In this case, the function returns the IVI_ERROR_REPEATED_CAPABILITY_NAME_REQUIRED error code.
- The repeated capability name refers to a specific repeated capability and the attribute does not apply to a repeated capability. In this case, the function returns the IVI_ERROR_REPEATED_CAPABILITY_NAME_NOT_ALLOWED error code.
- The repeated capability name refers to a specific repeated capability instance from which the attribute is restricted. In this case, the function returns the IVI_ERROR_ATTR_NOT_VALID_FOR_REPEATED_CAPABILITY error code.

Name	Туре	Description
vi	ViSession	The ViSession handle that you from Ivi_SpecificDriverNew. The identifies a particular IVI session
RepeatedCapabilityIdentifier	ViChar[]	Pass the repeated capability na you want to verify as valid for a particular attribute.
		The parameter accepts VI_NU empty string, a specific driver-(repeated capability identifier, o repeated capability identifier.
attributeID	ViAttr	Pass the ID of an attribute for t parameter.
		The include file for a specific in driver defines constant names the user-accessible attributes t to the driver. This includes attri that the IVI engine defines, attr that the instrument class define attributes that are specific to th particular instrument. Each def constant name begins with PREFIX_ATTR_, where PREFI2 specific instrument prefix.
		For each IVI engine attribute, t specific driver include file uses same constant name that appe ivi.h, except that the specific in prefix replaces the IVI prefix. F example, ivi.h defines IVI_ATTR_CACHE, and the Flu include file, fl45.h, defines the 1 constant name:

#define FL45_ATTR_CACHE IVI_ATTR_CACHE

For each instrument class attril specific driver include file uses same constant name that appe instrument class include file, ex the specific instrument prefix re the class prefix. For example, t class include file, ividmm.h, del IVIDMM_ATTR_RANGE, and t defines the following constant (

#define FL45_ATTR_RANGE IVIDMM_ATTR_RANGE

For each specific instrument at the specific driver include file d constant name and assigns a v is an offset from

IVI_SPECIFIC_PUBLIC_ATTR_ For example, fl45.h defines the constant name:

#define

FL45_ATTR_HOLD_THRESH (IVI_SPECIFIC_PUBLIC_ATT + 3L)

For each attribute that is privat instrument driver, the instrume source file defines a constant r assigns a value that is an offse IVI_SPECIFIC_PRIVATE_ATTR For example, hp34401a.c define following constant name:

#define

HP34401A_ATTR_TRIGGER_ (IVI_SPECIFIC_PRIVATE_AT + 1L)

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_ValidateRangeTable

Usage

ViStatus Ivi_ValidateRangeTable(IviRangeTablePtr Range Table);

Purpose

This function validates a range table. If you pass VI_NULL for the Range Table parameter, the function returns VI_SUCCESS. If you specify a non-NULL range table, the function returns an error if the range table type is not valid or the number of entries is zero.

The valid range table types are the following:

IVI_VAL_DISCRETE 0IVI_VAL_RANGED 1 IVI_VAL_COERCED 2

A range table has zero entries if the first entry has the following value in the cmdString field:

IVI_RANGE_TABLE_END_STRING ((ViString)(-1))

Name	Туре	Description
Range Table	lviRangeTablePtr	Specify the address of the range table you want to validate.
		Manual and a state NTT TO fair their and an atom

You can pass VI_NULL for this parameter.

Returns the type of the ViReal64 value you specify.

- (0) IVI_VAL_TYPE_NORMAL -normal value
- (1) IVI_VAL_TYPE_NAN -
 - -Not a Number (NaN)
- (2) IVI_VAL_TYPE_PINF
- -positive infinity
- (3) IVI_VAL_TYPE_NINF
- -negative infinity

Related Topic

Ivi_ValidateSession

Usage

ViStatus Ivi_ValidateSession(ViSession vi);

Purpose

This function checks an IVI session handle for validity. If the session is invalid, it returns an error code but does not set the primary error code, secondary error code, or error elaboration string for the current thread.

Name Type Description

vi ViSession The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_WriteFromFile

Usage

ViStatus Ivi_WriteFromFile(ViSession vi, ViChar filename[], ViInt32 writeNumberOfBytes, ViInt32 byteOffset, ViInt32* returnCount);

Purpose

This function reads data from a file you specify and writes it to an instrument using VISA I/O. Use this function internally in your instrument driver.

The function assumes that the IVI_ATTR_IO_SESSION attribute for the IVI session you specify holds a valid VISA session for the instrument.

The function opens the file in binary mode.

The function calls Ivi_SetNeedToCheckStatus with VI_TRUE.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
filename	ViChar[]	Specify the pathname of the file from which to write the data. You can specify an absolute pathname, a relative pathname, or a simple filename. The function treats relative pathnames and simple filenames as relative to the current working directory.
		If you enter a literal string in this parameter under Windows, be sure to use double backslashes to represent one backslash in the pathname.
writeNumberOfBytes	Vilnt32	Specify the maximum number of bytes to read from the file and write to the instrument.
byteOffset	Vilnt32	Specify the byte offset in the file at which to start reading.
		If the file contains header information that you do not want to write to the instrument, you can skip over the header by passing the number of bytes in the header for this parameter.
returnCount	Vilnt32*	Returns the number of bytes the function successfully writes from the file.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Ivi_WriteInstrData

Usage

ViStatus Ivi_WriteInstrData(ViSession vi, ViChar writeBuffer[]);

Purpose

This function writes a command string directly to an instrument using VISA I/O. The function bypasses the attribute state caching mechanism and therefore always invalidates all attribute cache values for the session. Use this function only to implement the PREFIX_WriteInstrData function that your instrument driver exports to the end-user.

The function assumes that the IVI_ATTR_IO_SESSION attribute for the IVI session you specify holds a valid VISA session for the instrument.

The function calls Ivi_SetNeedToCheckStatus with VI_TRUE.

Name	Туре	Description
vi	ViSession	The ViSession handle that you obtain from Ivi_SpecificDriverNew. The handle identifies a particular IVI session.
writeBuffer	ViChar[]	Specify the string you want to send to the instrument.

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

Function Tree, IVI Library

Class/Panel Name

Instrument Driver Session Create New Session Validate Session **Dispose Session** Locking Lock Session **Unlock Session** Channels **Build Channel Table** Add To Channel Table **Restrict Attr To Channels** Apply Default Setup Validate Attr For Channel **Coerce Channel Name** Get Channel Index Get Nth Channel String Get User Channel Name **Repeated Capabilities Build Rep Cap Table** Add To RepCap Table **Restrict Attr To Instances** Validate Attr For Rep Cap Name Coerce Rep Cap Name Get RepCap Index Get Nth Rep Cap String Attribute Creation

Function Name

Ivi_SpecificDriverNew Ivi_ValidateSession Ivi_Dispose

Ivi_LockSession Ivi_UnlockSession

Ivi_BuildChannelTableIvi_AddToChannelTableIvi_RestrictAttrToChannelsIvi_ApplyDefaultSetupIvi_ValidateAttrForChannelIvi_CoerceChannelNameIvi_GetChannelIndexIvi_GetNthChannelStringIvi_GetUserChannelName

Ivi_BuildRepCapTable Ivi_AddToRepCapTable Ivi_RestrictAttrToInstances Ivi_ValidateAttrForRepCapName

Ivi_CoerceRepCapName Ivi_GetRepCapIndex Ivi_GetNthRepCapString

Delete Attribute Add Attribute Add Attribute Vilnt32 Add Attribute Vilnt64 Add Attribute ViReal64 Add Attribute ViString Add Attribute ViBoolean Add Attribute ViSession Add Attribute ViAddr Add Repeated Attribute Add Repeat Attribute ViAddr Add Repeat Attribute ViBoolean Add Repeat Attribute Vilnt32 Add Repeat Attribute Vilnt64 Add Repeat Attribute ViReal64 Add Repeated Attribute ViSession Invalidation Lists Add Attribute Invalidation **Delete Attribute Invalidation Comparison Precision** Set Comparison Precision **Get Comparison Precision** Callbacks Set Range Table Callback Set Read Callback

Ivi_DeleteAttribute

- Ivi_AddAttributeViInt32
- Ivi_AddAttributeViInt64
- Ivi_AddAttributeViReal64
- Ivi_AddAttributeViString
- lvi_AddAttributeViBoolean
- Ivi_AddAttributeViSession
- Ivi_AddAttributeViAddr
- lvi_AddRepeatedAttributeViAddr
- Ivi_AddRepeatedAttributeViBoolean
- Ivi_AddRepeatedAttributeViInt32
- Ivi_AddRepeatedAttributeViInt64
- Ivi_AddRepeatedAttributeViReal64
- Ivi_AddRepeatedAttributeViSession
- Add Repeat Attribute ViString Ivi_AddRepeatedAttributeViString
 - Ivi_AddAttributeInvalidation
 - Ivi_DeleteAttributeInvalidation
 - Ivi_SetAttrComparePrecision
 - Ivi_GetAttrComparePrecision
 - Ivi_SetAttrRangeTableCallback

Set Read Callback Vilnt32 Set Read Callback Vilnt64 Set Read Callback ViReal64 Set Read Callback ViString Set Read Callback ViBoolean Set Read Callback ViSession Set Read Callback ViAddr Set Write Callback

Set Write Callback ViInt32 Set Write Callback ViInt64 Set Write Callback ViReal64 Set Write Callback ViString Set Write Callback ViBoolean Set Write Callback ViSession Set Write Callback ViAddr

Set Check Callback

Set Check Callback Vilnt32 Set Check Callback Vilnt64 Set Check Callback ViReal64 Set Check Callback ViString Set Check Callback

ViBoolean

Set Check Callback ViSession

Set Check Callback ViAddr Set Coerce Callback

Set Coerce Callback Vilnt32 Set Coerce Callback Vilnt64 Set Coerce Callback

- lvi_SetAttrReadCallbackViInt32
- Ivi_SetAttrReadCallbackViInt64
- lvi_SetAttrReadCallbackViReal64
- lvi_SetAttrReadCallbackViString
- lvi_SetAttrReadCallbackViBoolean
- Ivi_SetAttrReadCallbackViSession
- Ivi_SetAttrReadCallbackViAddr
- lvi_SetAttrWriteCallbackViInt32
- Ivi_SetAttrWriteCallbackViInt64
- Ivi_SetAttrWriteCallbackViReal64
- Ivi_SetAttrWriteCallbackViString
- Ivi_SetAttrWriteCallbackViBoolean
- Ivi_SetAttrWriteCallbackViSession
- Ivi_SetAttrWriteCallbackViAddr
- Ivi SetAttrCheckCallbackViInt32
- Ivi_SetAttrCheckCallbackViInt64
- Ivi_SetAttrCheckCallbackViReal64
- Ivi_SetAttrCheckCallbackViString
- lvi_SetAttrCheckCallbackViBoolean

lvi_SetAttrCheckCallbackViSession

lvi_SetAttrCheckCallbackViAddr

Ivi_SetAttrCoerceCallbackViInt32 Ivi_SetAttrCoerceCallbackViInt64

- Ivi_SetAttrCoerceCallbackViReal64

ViReal64

Set Coerce Callback ViString	lvi SetAttrCoerceCallbackViString
Set Coerce Callback	lvi_SetAttrCoerceCallbackViBoolean
ViBoolean	
Set Coerce Callback	<pre>lvi_SetAttrCoerceCallbackViSession</pre>
ViSession	
Set Coerce Callback ViAddr	lvi_SetAttrCoerceCallbackViAddr
Set Compare Callback	
Set Compare Callback ViInt32	Ivi_SetAttrCompareCallbackViInt32
Set Compare Callback ViInt64	lvi_SetAttrCompareCallbackViInt64
Set Compare Callback ViReal64	Ivi_SetAttrCompareCallbackViReal64
Set Compare Callback	Ivi SetAttrCompareCallbackViString
ViString	
Set Compare Callback	lvi_SetAttrCompareCallbackViBoolean
ViBoolean	
Set Compare Callback ViSession	Ivi_SetAttrCompareCallbackViSession
Set Compare Callback ViAddr	lvi_SetAttrCompareCallbackViAddr
Set/Get/Check Attribute	
Set Attribute	
Set Attribute Vilnt32	Ivi_SetAttributeViInt32
Set Attribute Vilnt64	Ivi_SetAttributeViInt64
Set Attribute ViReal64	Ivi_SetAttributeViReal64
Set Attribute ViString	Ivi_SetAttributeViString
Set Attribute ViBoolean	lvi_SetAttributeViBoolean
Set Attribute ViSession	Ivi_SetAttributeViSession
Set Attribute ViAddr	<u>lvi_SetAttributeViAddr</u>
Get Attribute	
Get Attribute Vilnt32	lvi_GetAttributeViInt32

Get Attribute Vilnt64 Get Attribute ViReal64 Get Attribute ViString Get Attribute ViBoolean Get Attribute ViSession Get Attribute ViAddr Check Attribute Check Attribute Vilnt32 Check Attribute Vilnt64 Check Attribute ViReal64 Check Attribute ViString Check Attribute ViBoolean Check Attribute ViSession Check Attribute ViAddr Caching/Status-Checking Control Invalidate Attribute Invalidate All Attributes Need To Check Status Set Need To Check Status Range Tables Get Attribute Range Table Validate Range Table Range Table Entries Get Vilnt32 Entry Vilnt32 Entry From Value Vilnt32 Entry From String ViInt32 Entry From Index

- Ivi_GetAttributeViInt64 Ivi_GetAttributeViReal64 Ivi_GetAttributeViString Ivi_GetAttributeViBoolean Ivi_GetAttributeViSession Ivi_GetAttributeViAddr
- Ivi CheckAttributeViInt32
- Ivi_CheckAttributeViInt64
- Ivi_CheckAttributeViReal64
- lvi_CheckAttributeViString
- lvi_CheckAttributeViBoolean
- lvi_CheckAttributeViSession
- lvi_CheckAttributeViAddr
- Ivi_InvalidateAttribute Ivi_InvalidateAllAttributes Ivi_NeedToCheckStatus Ivi_SetNeedToCheckStatus
- Ivi_GetAttrRangeTable Ivi_ValidateRangeTable
- Get Range Table Num Entries Ivi_GetRangeTableNumEntries
 - lvi_GetViInt32EntryFromValue
 - lvi_GetViInt32EntryFromString
 - lvi_GetViInt32EntryFromIndex

ViInt32 Entry From CmdValue Vilnt32 Entry From CoercedVal Get Vilnt64 Entry Vilnt64 Entry From Value Vilnt64 Entry From String Vilnt64 Entry From Index Vilnt64 Entry From CmdValue Vilnt64 Entry From CoercedVal Get ViReal64 Entry ViReal64 Entry From Value ViReal64 Entry From String ViReal64 Entry From Index ViReal64 Entry From CmdValue ViReal64 Entry From CoercedVal **Range Table Ptr** Get Stored Range Table Ptr Set Stored Range Table Ptr Dynamic Range Tables Range Table New Set Range Table Entry Set Range Table End Range Table Free Error Information Get Error Info

lvi_GetViInt32EntryFromCmdValue

lvi_GetViInt32EntryFromCoercedVal

lvi_GetViInt64EntryFromValue

lvi_GetViInt64EntryFromString

lvi_GetViInt64EntryFromIndex

lvi_GetViInt64EntryFromCmdValue

lvi_GetViInt64EntryFromCoercedVal

- Ivi_GetViReal64EntryFromValue
- Ivi_GetViReal64EntryFromString
- Ivi_GetViReal64EntryFromIndex

Ivi_GetViReal64EntryFromCmdValue

lvi_GetViReal64EntryFromCoercedVal

lvi_GetStoredRangeTablePtr

lvi_SetStoredRangeTablePtr

Ivi_RangeTableNew Ivi_SetRangeTableEntry Ivi_SetRangeTableEnd Ivi_RangeTableFree

lvi_GetErrorInfo

Get Error Message	<u>lvi_GetErrorMessage</u>
Get Specific Driver Status Desc	Ivi_GetSpecificDriverStatusDesc
Set Error Info	<u>lvi_SetErrorInfo</u>
Clear Error Info	<u>lvi_ClearErrorInfo</u>
Instrument Specific Error Queue	
Queue Instr Specific Error	<pre>lvi_QueueInstrSpecificError</pre>
Dequeue Instr Specific Error	<pre>lvi_DequeueInstrSpecificError</pre>
Clear Instr Specific Err Queue	Ivi_ClearInstrSpecificErrorQueue
Instr Specific Error Queue Size	Ivi_InstrSpecificErrorQueueSize
Memory Allocation	
Allocate Memory	<u>lvi_Alloc</u>
Free Allocated Memory	<u>lvi_Free</u>
Free All Allocated Memory	<u>lvi_FreeAll</u>
Helper Functions	
Get Info From Resource Name	Ivi_GetInfoFromResourceName
Inherent Attribute Accessors	
I/O Session	Ivi_IOSession
Range Checking	Ivi_RangeChecking
Query Instr Status	<u>lvi_QueryInstrStatus</u>
Simulating	<u>lvi_Simulating</u>
Use Specific Simulation	Ivi_UseSpecificSimulation
Spying	Ivi_Spying
Interchange Checking	Ivi_InterchangeCheck
String Callbacks	
Set Value in String Callback	Ivi_SetValInStringCallback
Direct Instrument I/O	

Write Instr Data Read Instr Data Read To File Write From File String/Value Tables Get String From Table Get Value From Table Value Manipulation Check Numeric Range Check Boolean Range **Coerce Boolean** Compare With Precision Get ViReal64 Type Default Callbacks Dflt Check Callback Vilnt32 Dflt Coerce Callback Vilnt32 Dflt Check Callback ViInt64 Dflt Coerce Callback Vilnt64 Dflt Check Callback ViReal64 Dflt Coerce Callback ViReal64 Dflt Compare Callback ViReal64 Dflt Coerce Callback ViBoolean Attribute Information Get Num Attributes Get Nth Attribute Get Attribute Name

Ivi_WriteInstrData Ivi_ReadInstrData Ivi_ReadToFile Ivi_WriteFromFile

Ivi_GetStringFromTable Ivi_GetValueFromTable

Ivi_CheckNumericRange

lvi_CheckBooleanRange

Ivi_CoerceBoolean

Ivi_CompareWithPrecision

Ivi_GetViReal64Type

lvi_DefaultCheckCallbackViInt32

Ivi_DefaultCoerceCallbackViInt32

lvi_DefaultCheckCallbackViInt64

lvi_DefaultCoerceCallbackViInt64

lvi_DefaultCheckCallbackViReal64

lvi_DefaultCoerceCallbackViReal64

Ivi_DefaultCompareCallbackViReal64

Ivi_DefaultCoerceCallbackViBoolean

Ivi_GetNumAttributes Ivi_GetNthAttribute Ivi_GetAttributeName
Set Attribute Flags Get Attribute Type Get Attribute Flags Get Invalidation List **Dispose Invalidation List** Attribute Is Cached Get Next Coercion Info Get Next Coercion String Get Attr Min Max Vilnt32 Get Attr Min Max Vilnt64 Get Attr Min Max ViReal64 Interchangeability Warnings Get Next Interchange Chk String **Clear Interchange Warnings Reset Interchange Check** Logical Names Get Logical Names List Get Nth Logical Name **Dispose Logical Names List** Configuration Get Config Store Handle Attach To Config Store Handle

- Ivi_SetAttributeFlags
- Ivi_GetAttributeType
- Ivi_GetAttributeFlags
- Ivi_GetInvalidationList
- Ivi_DisposeInvalidationList
- Ivi_AttributeIsCached
- Ivi_GetNextCoercionInfo
- lvi_GetNextCoercionString
- lvi_GetAttrMinMaxViInt32
- Ivi_GetAttrMinMaxViInt64
- lvi_GetAttrMinMaxViReal64
- Ivi_GetNextInterchangeCheckString
- lvi_ClearInterchangeWarnings
- lvi_ResetInterchangeCheck
- Ivi_GetLogicalNamesList
- Ivi_GetNthLogicalName
- Ivi_DisposeLogicalNamesList

Ivi_GetConfigStoreHandle AttachToConfigStoreHandle

IVI Library Overview

This help file describes the functions in the IVI engine. Also included are IVI Status Codes and error reporting information. Use the IVI engine to create VXI*plug&play* instrument drivers with advanced features, such as state caching, simulation, and compatibility with generic instrument classes.

The IVI Instrument Driver Wizard supplements the library, automatically creating the skeleton of an IVI driver that includes source code and function panels. The *LabWindows/CVI Instrument Driver Developers Guide* contains the IVI engine function reference information and instructions on how to create IVI drivers. Select **ToolsCreate IVI Specific Driver** to start the IVI Instrument Driver Wizard.

Refer to <u>IVI Library Function Tree</u> or click the function name in the *Alphabetical List of IVI Functions* section of the Contents tab for more information.

Related Documentation

LabWindows/CVI Instrument Driver Developers Guide—This document describes guidelines for writing an IVI instrument driver. To download and use this document, click <u>ni.com/manuals</u>, and search for the LabWindows/CVI Instrument Driver Developers Guide.

IVI Status Codes

IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Status Code Tables

Status Code Ranges

Default Values of Defined Constants

IVI Errors and Warnings

Common Instrument Driver Errors and Warnings

Most-Often-Encountered VISA Errors and Warnings

The following table defines the different ranges of status codes. The table lists the include files that contain the defined constants for the particular status codes.

Status Code Ranges

Status Code Type	Numeric Range (in Hex)	Include File
IVI Errors	BFFA0000 to BFFA1FFF	ivi.h
IVI Warnings	3FFA0000 to 3FFA1FFF	ivi.h
Class Driver Errors	BFFA2000 to BFFA3FFF	IviClass.h
Class Driver Warnings	3FFA2000 to 3FFA3FFF	IviClass.h
Specific Driver Errors	BFFA4000 to BFFA5FFF	Prefix.h
Specific Driver Warnings	3FFA4000 to 3FFA5FFF	Prefix.h
Common Instrument Driver Errors	BFFC0000 to BFFCFFFF	vpptype.h
Common Instrument Driver Warnings	3FFC0000 to 3FFCFFFF	vpptype.h
VISA Errors	BFFF0000 to BFFFFFFF	visa.h
VISA Warnings	3FFF0000 to 3FFFFFFF	visa.h

The Common Errors and Warnings are values that VXI*plug&play* defines and that specific instrument drivers return. They provide a consistent set of codes for error and warning conditions that are common among all instrument drivers. Each particular instrument driver defines its own set of Driver Errors and Warnings. The status codes values for one driver can overlap the status code values for other drivers.

The IVI engine and instrument driver include files define particular status codes as the unsigned sum of a base value and a decimal integer value. The following table lists the base values.

Default Values of Defined Constants

Status Code Type	Defined Constant for Base Value	Value
IVI Errors	IVI_ERROR_BASE	BFFA0000
IVI Warnings	IVI_WARN_BASE	3FFA0000
Class Driver Errors	IVI_CLASS_ERROR_BASE	BFFA2000
Class Driver Warnings	IVI_CLASS_WARN_BASE	3FFA2000
Specific Driver Errors	IVI_SPECIFIC_ERROR_BASE	BFFA4000
Specific Driver Warnings	IVI_SPECIFIC_WARN_BASE	3FFA4000

For example, if you pass an invalid attribute ID to an IVI engine function, the function returns IVI_ERROR_INVALID_ATTRIBUTE, which ivi.h defines as IVI_ERROR_BASE + 12, or 0xBFFA000C.

The following tables contain the IVI Status Codes, the Common Status Codes, and the most commonly used VISA Status Codes.

IVI Errors and Warnings

Status Description

- 0 The call was successful.
- BFFA0000 Unrecoverable failure.
- BFFA0001 Instrument error detected. Call the <u>*ClassPrefix*</u> error_query function and examine the error.
- BFFA0002 File could not be opened.
- BFFA0003 File is being read.
- BFFA0004 File is being modified.
- BFFA0005 Driver module file not found.
- BFFA0006 Cannot open driver module file for reading.
- BFFA0007 Driver module has invalid file format or contains invalid data.
- BFFA0008 Driver module contains undefined references.
- BFFA0009 Cannot find function in driver module.
- BFFA000A Failure loading driver module.
- BFFA000B The path name is invalid.
- BFFA000C Attribute ID not recognized.
- BFFA000D Attribute is read-only.
- BFFA000E Attribute is write-only.
- BFFA000F Invalid parameter.
- BFFA0010 Invalid value for parameter or property.
- BFFA0011 Function or method not supported.
- BFFA0012 Attribute or property not supported.
- BFFA0013 The enumeration value for the parameter is not supported.
- BFFA0014 Invalid Type.
- BFFA0015 The attribute and function parameter types do not match.
- BFFA0016 The specified attribute already has a value waiting to be updated.
- BFFA0017 The specified item already exists.

BFFA0018 Not a valid configuration.

- BFFA0019 The requested item or value does not exist or is not available.
- BFFA001A The requested attribute value not known and cannot be determined.
- BFFA001B There is no range table for this attribute.
- BFFA001C The range table is invalid.
- BFFA001D A connection to the instrument has not been initialized.
- BFFA001E The class instrument driver has encountered underspecified instrument configurations that limit interchangeability.
- BFFA001F No channel table has been built for the session. The instrument driver must call Ivi_BuildChannelTable in its IviInit function.
- BFFA0020 Channel or repeated capability name specified is not valid for the instrument.
- BFFA0021 Unable to allocate system resource.
- BFFA0022 Permission to access file was denied.
- BFFA0023 Too many files opened.
- BFFA0024 Unable to create temporary file in target directory.
- BFFA0025 All temporary filenames already used in target directory.
- BFFA0026 Disk is full.
- BFFA0027 Configuration file was not found 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 could not be loaded because of an 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 one or more required functions.
- BFFA0034 CVI Run-Time Engine initialization failed, probably because of insufficient memory.
- BFFA0035 CVI Run-Time Engine could not be loaded because of an unresolved external reference.
- BFFA0036 Failure loading CVI Run-Time Engine.
- BFFA0037 Cannot open DLL to read exports.
- BFFA0038 DLL file is corrupt.
- BFFA0039 No export table in DLL.
- BFFA003A Unknown attribute name for initial setting in configuration file.
- BFFA003B Invalid attribute value for initial setting in configuration file.
- BFFA003C Memory pointer specified is not known.
- BFFA003D Unable to find any channel or repeated capability strings.
- BFFA003E The channel or repeated capability list contains two instances of the same name.
- BFFA003F The VirtualChannelNames item in the configuration file contains a duplicate virtual channel name.
- BFFA0040 The VirtualChannelNames item in the configuration file contains an entry without a virtual channel name (nothing before the '=').
- BFFA0041 The VirtualChannelNames item in the configuration file contains an invalid virtual channel name. Channel names must contain only alphanumerics, underscores, or an exclamation point.
- BFFA0042 The VirtualChannelNames item in the configuration file contains a virtual channel name without an assigned channel string (i.e., nothing after '=').

- BFFA0043 The VirtualChannelNames item in the configuration file contains a virtual channel name that is assigned to an unknown or invalid channel string.
- BFFA0044 Channel or repeated capability name required.
- BFFA0045 The channel or repeated capability name is not allowed.
- BFFA0046 The attribute is not valid for the specified channel or repeated capability.
- BFFA0047 This operation requires a channel– or repeated capability– based attribute. The specified attribute is not channel– or repeated capability–based.
- BFFA0048 The channel has already been excluded for the specified attribute and cannot be re-included.
- BFFA0049 The option string parameter contains an entry without a name.
- BFFA004A The option string parameter contains an entry without a value.
- BFFA004B The option string parameter contains an entry with an unknown option name.
- BFFA004C The option string parameter contains an entry with an unknown option value.
- BFFA004D This operation is valid only on a sesssion created by a class driver.
- BFFA004E You cannot create a configuration file named "ivi.ini". That filename is reserved.
- BFFA004F There already is an entry of the same name in the run-time configuration.
- BFFA0050 The index parameter is one-based. You must pass a number greater than or equal to 1.
- BFFA0051 The index exceeds the number of items available.
- BFFA0052 You cannot set the cache for an attribute that has the IVI_VAL_NEVER_CACHE flag.
- BFFA0053 An instrument driver cannot export a ViAddr attribute to the end-user. Use the IVI_VAL_HIDDEN flag macro to make it a

private attribute.

- BFFA0054 Channel or repeated capability strings must contain only alphanumerics, underscores, or an exclamation point.
- BFFA0055 The Prefix item in the configuration file does not match the specific driver's prefix
- BFFA0056 The necessary memory could not be allocated.
- BFFA0057 Operation in progress.
- BFFA0058 NULL pointer passed for parameter or property.
- BFFA0059 Unexpected response from the instrument.
- BFFA005B File not found.
- BFFA005C The file format is invalid.
- BFFA005D The instrument status is not available.
- BFFA005E Instrument ID Query failed.
- BFFA005F Instrument reset failed.
- BFFA0060 Insufficient location information or resource not present in the system.
- BFFA0061 The driver is already initialized.
- BFFA0062 The simulation state cannot be changed.
- BFFA0063 Invalid number of levels in selector.
- BFFA0064 Invalid range in selector.
- BFFA0065 Unknown name in selector.
- BFFA0066 Badly-formed selector.
- BFFA0067 Unknown physical selector.
- BFFA1190 The session handle is not valid.
- BFFA1198 The session handle is not valid.
- BFFA11A0 Could not create thread local.
- BFFA1200 The specified configuration store file could not be deserialized.
- BFFA1201 A deserialize was attempted after a previous deserialize had already succeeded.

- BFFA1202 The specified configuration store file could not be serialized.
- BFFA1203 The session name or logical name could not be resolved to a session or driver session.
- BFFA1204 The item does not exist in the global collection.
- BFFA1205 An entry with name already exists in the collection.
- BFFA1206 The registry entry for the master configuration store does not exist or the file could not be found.
- BFFA1207 The item does not exist in the collection.
- BFFA1208 The data component is not a valid data component.
- BFFA1209 The element cannot be removed from the global collection when it is referenced in the local collections.
- BFFA1232 The specified handle is invalid or of an incorrect type.
- BFFA1233 The specified property ID is not valid for this function.
- BFFA6000 Repeated Capability lists cannot be modified after attributes have been added to them.
- BFFA6001 An attribute can only be restricted to a subset of a repeated capability once.
- BFFA6002 The repeated capability table cannot be built because it already exists.
- BFFA6003 The repeated capability has not been defined yet.
- BFFA6004 The repeated capability name cannot be an empty or NULL string.
- BFFA600D The Config Server module is not present on the system.
- 3FFA0065 Identification query not supported.
- 3FFA0066 Reset operation not supported.
- 3FFA0067 Self test operation not supported.
- 3FFA0068 Error query operation not supported.
- 3FFA0069 Revision query not supported.

Common Instrument Driver Errors and Warnings

Status Description

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.
BFFC0010 Instrument failed the ID Query.
BFFC0101 Instrument does not have ID Query 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.

Most-Often-Encountered VISA Errors and Warnings

Status Description

- BFFF0000 Miscellaneous or system error occurred.
- BFFF000E Invalid session handle.
- BFFF0015 Timeout occurred before operation could complete.
- BFFF0034 Violation of raw write protocol occurred.
- BFFF0035 Violation of raw read protocol occurred.
- BFFF0036 Device reported an output protocol error.
- BFFF0037 Device reported an input protocol error.
- BFFF0038 Bus error occurred during transfer.
- BFFF003A Invalid setup (attributes are not consistent).
- BFFF005F No listeners condition was detected.
- BFFF0060 This interface is not the controller in charge.
- BFFF0067 Operation is not supported on this session.
- 3FFF0085 The status value you passed is unknown.

Required Functions

Expand this book to view an alphabetized list of required IVI functions.

Required IVI Functions Overview

This topic contains information and function descriptions for instrument driver functions that each IVI instrument driver must implement. A related LabWindows/CVI Instrument Driver Developers Guide exists to help you develop and use instrument drivers. To download and use this document, click <u>ni.com/manuals</u>, and search for the LabWindows/CVI Instrument Driver Developers Guide.

The required functions are grouped into the following three categories. To access information about each function, you can use the following list of required functions or click on *Alphabetical List of Required IVI Functions* section of the Contents tab.

Initialize/Close Functions

- <u>Prefix_init</u>
- <u>Prefix</u>_InitWithOptions
- <u>Prefix_close</u>
- <u>Prefix</u>_IviInit
- <u>Prefix_IviClose</u>

Utility Functions

- <u>Prefix_self_test</u>
- <u>Prefix</u>_Disable
- <u>Prefix</u>_reset
- **<u>Prefix</u>**<u>ResetWithDefaults</u>
- **<u>Prefix</u>** revision_query
- <u>Prefix_error_query</u>
- <u>Prefix_error_message</u>

Wrappers for IVI engine Functions

- Prefix _GetError
- *Prefix* _ClearError
- *Prefix* _LockSession
- Prefix _UnlockSession
- *Prefix* _ReadInstrData
- Prefix _WriteInstrData
- *Prefix* _GetNextInterchangeWarning

- *Prefix* _ClearInterchangeWarnings
- *Prefix* _ResetInterchangeCheck
- *Prefix* _InvalidateAllAttributes
- *Prefix* _GetNextCoercionRecord

Except for *Prefix*_IviInit and *Prefix*_IviClose, all the required functions are user-callable. The instrument driver must have function panels for the user-callable functions and must have function prototypes for them in the include file. No function panels or prototypes are necessary for the *Prefix*_IviInit and *Prefix*_IviClose functions.

The *Prefix*_init function and the *Prefix*_InitWithOptions function call *Prefix*_IviInit. The *Prefix*_close function calls *Prefix*_IviClose. In a specific instrument driver, the *Prefix*_IviInit and *Prefix*_IviClose functions contain the bulk of the code that performs the initialization and closing operations for the particular instrument.

This section contains descriptions for the Initialize/Close functions and the Utility functions, including the implementation requirements. For the Wrapper functions, refer to the corresponding functions in the <u>IVI engine</u> <u>Overview</u>.

Note If you use the instrument driver developer wizard to create your instrument driver, the source file that the wizard generates contains the required functions with as much of the implementation source code as possible and with comments explaining how to complete the functions.

Prefix_Disable

ViStatus Prefix_Disable (ViSession vi);

Purpose

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

Parameter

Input

Name Type Description

vi ViSession Unique identifier for an IVI session.

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Implementation Requirements

Call Ivi_LockSession to lock the IVI session.

If simulation is disabled, send the appropriate commands to return the instrument to a quiescent state. Remember to call Ivi_SetNeedToCheckStatus with VI_TRUE before you perform the instrument I/O.

Finally, call Ivi_UnlockSession to unlock the IVI session.

Be sure to document the state in which the Prefix_Disable function places the instrument. Include the information in the function panel help for the Prefix_Disable function.

*Prefix_*init

ViStatus *Prefix_*init (ViRsrc resourceName, ViBoolean IDQuery, ViBoolean resetDevice, ViSession *instrumentHandle);

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 Prefix_LockSession and Prefix_UnlockSession functions to protect sections of code that require exclusive access to the resource.

Parameters

Name	Туре	Description
resourceName	ViRsrc	Pass the resource name of the device to initialize. You can also pass the name of a driver session or logical name that you configure with the IVI Configuration utility. The driver session identifies a specific device and specifies the initial settings for the session. A logical Name identifies a particular driver session. Refer to the <i>Parameter Discussion</i>

		section for more information on resourceName.
IDQuery	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	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
instrument Handle	ViSession (passed by reference)	 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. This is useful if you have multiple physical instances of the same type of instrument. Note 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 Prefix_LockSession and Prefix_UnlockSession to protect sections of code that require exclusive access to the resource.

Parameter Discussion

Refer to the following for the exact grammar to use for the **resourceName** parameter. Optional fields are shown in square brackets ([]).

Syntax

GPIB[board]::<primary address>[::secondary address]::INSTR VXI VXI[board]::VXI logical address[::INSTR] GPIB-VXI GPIB-VXI[board][::GPIB-VXI primary address] ::VXI logical address[::INSTR] Serial ASRL<port>::INSTR <LogicalName>

Optional Field – Value

If you do not specify a value for an optional field, the following values are used:

board - 0

secondary address - none (31)

Valid Value	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
"VXI::64::INSTR"	VXI board 0, logical address 64
"VXI1::64::INSTR"	VXI board 1, logical address 64
"GPIB-	GPIB-VXI board 0, logical address 64

VXI::64::INSTR"	
"GPIB- VXI1::64::INSTR"	GPIB-VXI board 1, logical address 64
"ASRL2::INSTR"	COM port 2
"SampleInstr"	Logical name "SampleInstr"
"Prefix"	Logical Name or Driver Session "Prefix"
"Prefix"	Logical Name or Driver Session "Prefix

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Prefix_InitWithOptions

ViStatus *Prefix*_InitWithOptions (ViRsrc resourceName, ViBoolean IDQuery, ViBoolean resetDevice, ViString optionsString, ViSession *instrumentHandle);

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:
 PREFIX_ATTR_INTERCHANGE_CHECK
 PREFIX_ATTR_RANGE_CHECK
 PREFIX_ATTR_QUERY_INSTRUMENT_STATUS
 PREFIX_ATTR_CACHE
 PREFIX_ATTR_SIMULATE
 PREFIX_ATTR_RECORD_COERCIONS
 PREFIX_ATTR_DRIVER_SETUP
- 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 Prefix_LockSession and
Prefix_UnlockSession functions to protect sections of code that
require exclusive access to the resource.
```

Parameters

Name Type Description

resourceName	ViRsrc	Pass the resource name of the device to initialize. You can also pass the name of a driver session or logical name that you configure with the IVI Configuration utility. The driver session identifies a specific device and specifies the initial settings for the session. A logical Name identifies a particular driver session. Refer to the <i>Parameter Discussion</i> section for more information on resourceName .
IDQuery	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	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
optionsString	ViString	You can use this control to set the initial value of certain attributes for the session. Refer to the <i>Parameter Discussion</i> for more information on optionsString .
instrument	ViSession	Returns a ViSession handle that you use to

Handle	(passed by	identify the	instrument in all subsequent driver function calls.
	reference)	Note	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.

Note 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 Prefix_LockSession and Prefix_UnlockSession to protect sections of code that require exclusive access to the resource.

Parameter Discussion

Using the resourceName Parameter

Refer to the following table below for the exact grammar to use for the **resourceName** parameter. Optional fields are shown in square brackets ([]).

Syntax

GPIB[board]::<primary address>[::secondary address]::INSTR VXI VXI[board]::VXI logical address[::INSTR] GPIB-VXI GPIB-VXI[board][::GPIB-VXI primary address] ::VXI logical address[::INSTR]

Serial ASRL<port>::INSTR

<LogicalName>

Optional Field and Value

If you do not specify a value for an optional field, the following values are used:

board - 0 secondary address - none (31)

Valid Value	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
"VXI::64::INSTR"	VXI board 0, logical address 64
"VXI1::64::INSTR"	VXI board 1, logical address 64
"GPIB- VXI::64::INSTR"	GPIB-VXI board 0, logical address 64
"GPIB- VXI1::64::INSTR"	GPIB-VXI board 1, logical address 64
"ASRL2::INSTR"	COM port 2
"SampleInstr"	Logical name "SampleInstr"
"Prefix"	Logical Name or Driver Session "Prefix"

Using the optionsString Parameter

The following table lists the attributes and the name you use in the **optionsString** parameter to identify the attribute.

Name in optionsString	Attribute Defined Constant
RangeCheck	PREFIX_ATTR_RANGE_CHECK
QueryInstrStatus	PREFIX_ATTR_QUERY_INSTRUMENT_STATUS
Cache	PREFIX_ATTR_CACHE
Simulate	PREFIX_ATTR_SIMULATE
RecordCoercions	PREFIX_ATTR_RECORD_COERCIONS
InterchangeCheck	PREFIX_ATTR_INTERCHANGE_CHECK
DriverSetup	PREFIX_ATTR_DRIVER_SETUP

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.

The session uses the values you specify in the **optionsString** parameter. If you pass a NULL or an empty string for the **optionsString** parameter, and you pass either a DriverSession or logicalName to the **resourceName** parameter, the session uses the values specified in the IVI Configuration file. If you do not specify an attribute value in either the **optionsString** parameter or in the IVI Configuration file, the default value for the attribute is used.

The default values for the attributes are shown below:

Attribute Name Default Value

VI_TRUE
VI_FALSE
VI_TRUE
VI_FALSE
VI_FALSE

InterchangeCheck VI_FALSE

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=0,Cache=1"

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

*Prefix*_IviInit

ViStatus *Prefix_*IviInit (ViRsrc resourceName, ViBoolean idQuery, ViBoolean reset, ViSession vi);

Purpose

Contains the bulk of the code to initialize an instrument driver session. *Prefix_*InitWithOptions calls *Prefix_*IviInit after it calls Ivi_SpecificDriverNew to create the IVI session

Note Refer to *Prefix_*init for a description of the initialization actions and parameters. This function description documents only the implementation requirements for *Prefix_*IviInit.

Implementation Requirements

Call Ivi_BuildChannelTable to specify the set of valid channel strings for the instrument. If the instrument does not have multiple channels, specify "1" for the channel strings.

Create all of the attributes you want to use, excluding the inherent IVI attributes. If you are developing your driver according to a class definition, create the class attributes that you want to use. Generally, you should create the attributes in an internal *Prefix*_InitAttributes function. If you use the instrument driver developer wizard, the wizard creates this function for you. You must have this function in your source file to be able to use the attribute editor.

If simulation is disabled, create the I/O session, and set the IVI_ATTR_IO_SESSION attribute. Configure the I/O interface.

If the **reset** parameter is VI_TRUE, call *Prefix*_reset. Otherwise, if not simulating, send the default setup commands to the instrument. It is best to have an internal *Prefix*_DefaultInstrSetup function for this purpose because *Prefix*_reset also sends the default setup commands.

If the **idQuery** parameter is VI_TRUE and simulation is disabled, verify the identity of the instrument if possible. For IEEE 488.2 compatible instruments, use the *IDN? query. For VXI register-based instruments, check the manufacturer ID and model number. Return an error if the instrument is not one that the driver supports.

Finally, check the status of the instrument. Typically, you do this by calling an internal *Prefix*_CheckStatus function. The instrument driver developer

wizard generates the internal *Prefix*_CheckStatus function for you. The internal *Prefix*_CheckStatus function calls the check status callback if status checking is enabled, simulation is disabled, and the driver has performed instrument I/O since the last time it queried the instrument status.

If a failure occurs after you open the I/O session, close the I/O session and set the IVI_ATTR_IO_SESSION parameter back to 0.

*Prefix_*close

ViStatus *Prefix_*close (ViSession vi);

Purpose

When you are finished using an instrument driver session, you must call the *Prefix_*close function. *Prefix_*close performs the following actions.

- Closes the instrument I/O session.
- Destroys the IVI session and all of its attributes.
- Deallocates any memory resources used by the IVI session.

You might also want to put the instrument into an idle state before closing the I/O session. For example, a switch driver might disconnect all switches.

Parameter

Input

Name Type Description

vi ViSession Unique identifier for an IVI session.

Implementation Requirements

First call Ivi_LockSession to lock the IVI session.

Call *Prefix*_IviClose.

Call Ivi_UnlockSession to unlock the IVI session.

Finally, call Ivi_Dispose on the IVI session. It is very important to unlock the IVI session *before* calling Ivi_Dispose.

Ivi_Dispose destroys the instrument driver session and all of its attributes. It also deallocates any memory blocks that you associated with the session when you called Ivi_Alloc or Ivi_RangeTableNew. *Prefix_*IviClose performs all the other clean-up operations.

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Prefix_IviClose

ViStatus Prefix_IviClose (ViSession vi);

Purpose

Performs all the clean-up operations for closing an instrument driver session, except for destroying the IVI session. The Prefix_close functions in the specific driver calls Prefix_IviClose.

Implementation Requirements

Set the instrument to an idle state, if that is appropriate.

Close the I/O session and set IVI_ATTR_IO_SESSION to 0.

If you have any hidden ViAddr attributes that point to memory that you dynamically allocated, free the memory.

Prefix_reset

ViStatus *Prefix_*reset (ViSession vi);

Purpose

Places the instrument in a known state. In an IEEE 488.2 instrument, the _reset function sends the command string "*RST" to the instrument. *Prefix_*reset also sends the default setup commands to the instrument to configure settings for the proper operation of the instrument driver. You can either call the *Prefix_*reset function separately, or you can select it to be called from the *Prefix_*init function.

Parameter

Input

Name Type Description

vi ViSession Unique identifier for an IVI session.

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Implementation Requirements

Call Ivi_LockSession to lock the IVI session.

If simulation is disabled, send the reset command to the instrument. On IEEE 488.2 instruments, you do this by sending the *RST command. Remember to call Ivi_SetNeedToCheckStatus with VI_TRUE before you perform the instrument I/O.

Send the default setup commands to the instrument. It is best to have an

internal *Prefix*_DefaultInstrSetup function for this purpose. *Prefix*_IviInit must call *Prefix*_DefaultInstrSetup when it does not call *Prefix*_Reset.

Finally, call Ivi_UnlockSession to unlock the IVI session.

Be sure to document the state in which the *Prefix*_reset function places the instrument. Include the information in the function panel help for the *Prefix*_reset function.

Prefix_self_test

ViStatus *Prefix_*self_test (ViSession vi, ViInt16 * testResult, ViChar testMessage[]);

Purpose

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

Parameter

Innut

Туре	Description
ViSession	Unique identifier for an IVI session.
Туре	Description
Vilnt16	Numeric result from self-test operation 0 = no error (test passed)
	Type ViSession Type Vilnt16

testMessage ViChar array Self-test status message

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Implementation Requirements

Report an error if the user passes VI_NULL for either of the output parameters.
Call Ivi_LockSession to lock the IVI session.

If simulation is disabled, send the self-test command to the instrument. On IEEE 488.2 instruments, you do this by sending the *TST command. Then read the results from the instrument into the **testResult** and **testMessage** output parameters.

If simulation is enabled but Ivi_UseSpecificSimulation returns VI_TRUE, set the **testResult** output parameter to 0, and copy "No error." into the **testMessage** output parameter.

Call your internal *Prefix*_CheckStatus function.

Finally, call Ivi_UnlockSession to unlock the IVI session.

If the instrument cannot perform a self-test operation, you should still include the function in the driver and return the warning VI_WARN_NSUP_SELF_TEST.

*Prefix_*error_query

ViStatus *Prefix_*error_query (ViSession vi, ViInt32 * errCode, ViChar errMessage[]);

Purpose

Queries the instrument and returns the instrument specific error information.

Generally, you call this function after another function in the instrument driver returns the IVI_ERROR_INSTRUMENT_STATUS error code. The driver returns IVI_ERROR_INSTRUMENT_STATUS when the instrument's status register indicates that the instrument's error queue is not empty. *Prefix_*error_query extracts the first error out of the instrument's error queue. For instruments that have status registers but no error queue, the driver simulates an error queue in software.

Parameter

Input

Name	Туре	Description
vi	ViSession	Unique identifier for an IVI session.
Output		
Name	Туре	Description
errCode	Vilnt32	Instrument error code

errMessage ViChar array Instrument error message

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Implementation Requirements

Report an error if the user passes VI_NULL for either of the output parameters.

If the instrument has status registers and an error queue, do the following.

- 1. Call Ivi_LockSession to lock the IVI session.
- 2. If simulation is disabled, send the error query command to the instrument. On IEEE 488.2 instruments, you do this by sending the :SYST:ERR? command. Then read the results from the instrument into the **errCode** and **errMessage** output parameters.
- 3. If simulation is enabled, set the **errCode** output parameter to 0, and copy "No error." into the **errCode** output parameter.
- 4. Call Ivi_UnlockSession to unlock the IVI session.

Some instruments have status registers but no error queue. The act of reading the status registers clears the error information. For such instruments, the check status callback must call Ivi_QueueInstrSpecificError to add the error information to a software error queue whenever the status registers indicate an error. The *Prefix_*error_query function must do the following:

- 1. Call Ivi LockSession to lock the IVI session.
 - 2. Call Ivi_InstrSpecificErrorQueueSize to determine if the software error queue is empty. If it is empty, call the check status callback and then Ivi_InstrSpecificErrorQueueSize again to determine if the software error queue is still empty.
 - 3. If the software queue is not empty, call Ivi_DequeueInstrSpecificError to extract the error information into the **errCode** and **errMessage** parameters.
 - 4. Otherwise, set the **errCode** output parameter to 0, and copy "No error." into the **errMessage** output parameter.
 - 5. Call Ivi_UnlockSession to unlock the IVI session.

If the instrument cannot perform an error query, you should still include the function in the driver and return the warning VI_WARN_NSUP_ERROR_QUERY.

*Prefix_*error_message

ViStatus *Prefix_*error_message (ViSession vi, ViStatus errCode, ViChar errMessage[]);

Purpose

Translates the error return value from an instrument driver function to a user-readable string.

Parameter

Input

Name	Туре	Description
vi	ViSession	Unique identifier for an IVI session. Can be VI_NULL
errCode	ViStatus	Instrument driver error code
Output		
Name	Туре	Description
errMessage	ViChar array	Instrument driver error message

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Implementation Requirements

Your *Prefix_*error_message function must accept a value of VI_NULL for the **vi** input parameter. This allows the user to call the function even

when *Prefix_*init or *Prefix_*InitWithOptions fails. On the other hand, report an error if the user passes VI_NULL for the address of the **errMessage** output buffer.

If your driver defines its own error codes, define a static string/value table containing the error codes and message strings. Use the IviStringValueTable typedef in ivi.h. Terminate the table with an entry that

has VI_NULL in both fields.

If the **vi** parameter is not VI_NULL, call Ivi_LockSession to lock the IVI session.

Call the Ivi_GetSpecificDriverStatusDesc function. Pass the address of your error string/value table as the last parameter. If your driver does not have its own error codes, pass VI_NULL for the last parameter.

If the **vi** parameter is not VI_NULL, call Ivi_UnlockSession to unlock the IVI session.

Prefix_ResetWithDefaults

ViStatus Prefix_ResetWithDefaults (ViSession vi);

Purpose

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 <u>Prefix_reset</u>.

Parameter

Input

Name Type Description

vi ViSession Unique identifier for an IVI session.

Implementation Requirements

Call Ivi_LockSession to lock the IVI session.

Call PREFIX_reset.

Call Ivi_ApplyDefaultSetup.

Finally, call Ivi_UnlockSession to unlock the IVI session.

Prefix_revision_query

ViStatus *Prefix_*revision_query (ViSession vi, ViChar driverRev[], ViChar instrRev[]);

Purpose

Obtains the following information:

- The revision of the instrument driver
- The firmware revision of the instrument you are currently using

Parameter

Input		
Name	Туре	Description
vi	ViSession	Unique logical identifier to a session with an instrument
Output		
Name	Туре	Description
driverRev	ViChar array	Instrument driver revision
instrRev	ViChar array	Instrument firmware revision

Return Values

Contains the status code that the function call returns. IVI engine functions can return error and warning values from several sets of status codes. Some status codes are unique to the IVI engine. Other status codes are the same codes that VISA Library functions return. Still others are error or warning values that functions in specific instrument drivers return. Each set of status codes has its own numeric range.

Regardless of the source of the status code, 0 always indicates success, a positive value indicates a warning, and a negative value indicates an error.

Related Topic

IVI Status Codes

Implementation Requirements

Report an error if the user passes VI_NULL for either of the output parameters.

Call Ivi_LockSession to lock the IVI session.

Call Ivi_GetAttributeViString on the IVI_ATTR_DRIVER_REVISION attribute to get the driver revision string into the **driverRev** output parameter.

If simulation is disabled, send the revision query command to the instrument. On IEEE 488.2 instruments, you do this by sending the *IDN command. Then read the results from the instrument into **instrRev** output parameter. Remember to call Ivi_SetNeedToCheckStatus with VI_TRUE before you perform the instrument I/O.

If simulation is enabled, copy "No revision information available while simulating." into the **instrRev** output parameter.

Call your internal *Prefix*_CheckStatus function.

Call Ivi_UnlockSession to unlock the IVI session.