Allied Vision

Vimba

Vimba

Vimba C Function Reference

1.9.1

In this chapter, you can find a complete list of all methods that are described in VimbaC.h.

All function and type definitions are designed to be platform-independent and portable from other languages.

General conventions:

- Method names are composed in the following manner:
  - Vmb"Action". Example: `VmbStartup()`
  - Vmb"Entity""Action". Example: `VmbInterfaceOpen()`
  - Vmb"ActionTarget""Action". Example: `VmbFeaturesList()`
  - Vmb"Entity""SubEntity""Action". Example: `VmbFeatureCommandRun()`
- Methods dealing with features, memory, or registers accept a handle from the following entity list as first parameter: System, Camera, Interface, and AncillaryData. All other methods taking handles accept only a specific handle.
- Strings (generally declared as "const char *") are assumed to have a trailing 0 character.
- All pointer parameters should of course be valid, except if stated otherwise.
- To ensure compatibility with older programs linked against a former version of the API, all struct* parameters have an accompanying sizeofstruct parameter.
- Functions returning lists are usually called twice: once with a zero buffer to get the length of the list, and then again with a buffer of the correct length.

Methods in this chapter are always described in the same way:

- The caption states the name of the function without parameters
- The first item is a brief description
- The parameters of the function are listed in a table (with type, name, and description)
- The return values are listed
- Finally, a more detailed description about the function is given

Callbacks

VmbInvalidationCallback

Invalidation Callback type for a function that gets called in a separate thread and has been registered with VmbFeatureInvalidationRegister()

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
in	void*	pUserContext	Pointer to the user context, see VmbFeatureInvalidationRegister

While the callback is run, all feature data is atomic. After the callback finishes, the feature data might be updated with new values.

Do not spend too much time in this thread; it will prevent the feature values from being updated from any other thread or the lower-level drivers.

VmbFrameCallback

Frame Callback type for a function that gets called in a separate thread if a frame has been queued with VmbCaptureFrameQueue()

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle of the camera
out	VmbFrame_t*	pFrame	Frame completed

API Version

VmbVersionQuery()

Retrieve the version number of VimbaC.

	Type	Name	Description
out	VmbVersionInfo_t*	pVersionInfo	Pointer to the struct where version information is copied
in	VmbUint32_t	sizeofVersionInfo	Size of structure in bytes

- **VmbErrorSuccess:** If no error
- **VmbErrorStructSize:** The given struct size is not valid for this version of the API
- **VmbErrorBadParameter:** If "pVersionInfo" is NULL.

This function can be called at anytime, even before the API is initialized. All other version numbers may be queried via feature access.

API Initialization

VmbStartup()

Initialize the VimbaC API.

- **VmbErrorSuccess:** If no error
- **VmbErrorInternalFault:** An internal fault occurred

On successful return, the API is initialized; this is a necessary call.

This method must be called before any VimbaC function other than VmbVersionQuery() is run.

VmbShutdown()

Perform a shutdown on the API.

This will free some resources and deallocate all physical resources if applicable.

Camera Enumeration & Information

VmbCamerasList()

Retrieve a list of all cameras.

	Type	Name	Description
out	VmbCameraInfo_t*	pCameraInfo	Array of VmbCameraInfo_t, allocated by the caller. The camera list is copied here. May be NULL if pNumFound is used for size query.
in	VmbUint32_t	listLength	Number of VmbCameraInfo_t elements provided
out	VmbUint32_t*	pNumFound	Number of VmbCameraInfo_t elements found.
in	VmbUint32_t	sizeofCameraInfo	Size of the structure (if pCameraInfo == NULL this parameter is ignored)

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorStructSize:** The given struct size is not valid for this API version
- **VmbErrorMoreData:** The given list length was insufficient to hold all available entries
- **VmbErrorBadParameter:** If "pNumFound" was NULL

Camera detection is started with the registration of the "DiscoveryCameraEvent" event or the first call of VmbCamerasList(), which may be delayed if no "DiscoveryCameraEvent" event is registered (see examples). VmbCamerasList() is usually called twice: once with an empty array to query the list length, and then again with an array of the correct length. If camera lists change between the calls, pNumFound may deviate from the query return.

VmbCameraInfoQuery()

Retrieve information on a camera given by an ID.

	Type	Name	Description
in	const char*	idString	ID of the camera
out	VmbCameraInfo_t*	pInfo	Structure where information will be copied. May be NULL.
in	VmbUint32_t	sizeofCameraInfo	Size of the structure

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorNotFound:** The designated camera cannot be found
- **VmbErrorStructSize:** The given struct size is not valid for this API version
- **VmbErrorBadParameter:** If "idString" was NULL

May be called if a camera has not been opened by the application yet. Examples for "idString": "DEV_81237473991" for an ID given by a transport layer, "169.254.12.13" for an IP address, "000F314C4BE5" for a MAC address or "DEV_1234567890" for an ID as reported by Vimba

VmbCameraOpen()

Open the specified camera.

	Type	Name	Description
in	const char*	idString	ID of the camera
in	VmbAccessMode_t	accessMode	Determines the level of control you have on the camera
out	VmbHandle_t*	pCameraHandle	A camera handle

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorNotFound:** The designated camera cannot be found
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorInvalidCall:** If called from frame callback
- **VmbErrorBadParameter:** If "idString" or "pCameraHandle" is NULL

A camera may be opened in a specific access mode, which determines the level of control you have on a camera. Examples for "idString": "DEV_81237473991" for an ID given by a transport layer, "169.254.12.13" for an IP address, "000F314C4BE5" for a MAC address or "DEV_1234567890" for an ID as reported by Vimba

VmbCameraClose()

Close the specified camera.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	A valid camera handle

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorInvalidCall:** If called from frame callback

Depending on the access mode this camera was opened with, events are killed, callbacks are unregistered, and camera control is released.

Features

VmbFeaturesList()

List all the features for this entity.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
out	VmbFeatureInfo_t*	pFeatureInfoList	An array of VmbFeatureInfo_t to be filled by the API. May be NULL if pNumFund is used for size query.
in	VmbUint32_t	listLength	Number of VmbFeatureInfo_t elements provided
out	VmbUint32_t*	pNumFound	Number of VmbFeatureInfo_t elements found. May be NULL if pFeatureInfoList is not NULL.
in	VmbUint32_t	sizeofFeatureInfo	Size of a VmbFeatureInfo_t entry

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorStructSize:** The given struct size of VmbFeatureInfo_t is not valid for this version of the API
- **VmbErrorMoreData:** The given list length was insufficient to hold all available entries

This method lists all implemented features, whether they are currently available or not. The list of features does not change as long as the camera/interface is connected. "pNumFound" returns the number of VmbFeatureInfo elements. This function is usually called twice: once with an empty list to query the length of the list, and then again with an list of the correct length.

VmbFeatureInfoQuery()

Query information about the constant properties of a feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	VmbFeatureInfo_t*	pFeatureInfo	The feature info to query
in	VmbUint32_t	sizeofFeatureInfo	Size of the structure

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorStructSize:** The given struct size is not valid for this version of the API

Users provide a pointer to VmbFeatureInfo_t, which is then set to the internal representation.

VmbFeatureListAffected()

List all the features that might be affected by changes to this feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	VmbFeatureInfo_t*	pFeatureInfoList	An array of VmbFeatureInfo_t to be filled by the API. May be NULL if pNumFound is used for size query.
in	VmbUint32_t	listLength	Number of VmbFeatureInfo_t elements provided
out	VmbUint32_t*	pNumFound	Number of VmbFeatureInfo_t elements found. May be NULL is pFeatureInfoList is not NULL.
in	VmbUint32_t	sizeofFeatureInfo	Size of a VmbFeatureInfo_t entry

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command

- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorStructSize:** The given struct size of VmbFeatureInfo_t is not valid for this version of the API
- **VmbErrorMoreData:** The given list length was insufficient to hold all available entries

This method lists all affected features, whether they are currently available or not. The value of affected features depends directly or indirectly on this feature (including all selected features). The list of features does not change as long as the camera/interface is connected. This function is usually called twice: once with an empty array to query the length of the list, and then again with an array of the correct length.

VmbFeatureListSelected()

List all the features selected by a given feature for this module.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	VmbFeatureInfo_t*	pFeatureInfoList	An array of VmbFeatureInfo_t to be filled by the API. May be NULL if pNumFound is used for size query.
in	VmbUint32_t	listLength	Number of VmbFeatureInfo_t elements provided
out	VmbUint32_t*	pNumFound	Number of VmbFeatureInfo_t elements found. May be NULL if pFeatureInfoList is not NULL.
in	VmbUint32_t	sizeofFeatureInfo	Size of a VmbFeatureInfo_t entry

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorStructSize:** The given struct size is not valid for this version of the API
- **VmbErrorMoreData:** The given list length was insufficient to hold all available entries

This method lists all selected features, whether they are currently available or not. Features with selected features ("selectors") have no direct impact on the camera, but only influence the register address that selected features point to. The list of features does not change while the camera/interface is connected. This function is usually called twice: once with an empty array to query the length of the list, and then again with an array of the correct length.

VmbFeatureAccessQuery()

Return the dynamic read and write capabilities of this feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features.
in	const char *	name	Name of the feature.
out	VmbBool_t *	pIsReadable	Indicates if this feature is readable. May be NULL.
out	VmbBool_t *	pIsWriteable	Indicates if this feature is writable. May be NULL.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorBadParameter:** If "pIsReadable" and "pIsWriteable" were both NULL
- **VmbErrorNotFound:** The feature was not found

The access mode of a feature may change. For example, if "PacketSize" is locked while image data is streamed, it is only readable.

Integer

VmbFeatureIntGet()

Get the value of an integer feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	VmbInt64_t*	pValue	Value to get

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Integer
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "name" or "pValue" is NULL

VmbFeatureIntSet()

Set the value of an integer feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
in	VmbInt64_t	value	Value to set

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Integer
- **VmbErrorInvalidValue:** If "value" is either out of bounds or not an increment of the minimum
- **VmbErrorBadParameter:** If "name" is NULL

- **VmbErrorNotFound:** If the feature was not found
- **VmbErrorInvalidCall:** If called from frame callback

VmbFeatureIntRangeQuery()

Query the range of an integer feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	VmbInt64_t*	pMin	Minimum value to be returned. May be NULL.
out	VmbInt64_t*	pMax	Maximum value to be returned. May be NULL.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorBadParameter:** If "name" is NULL or "pMin" and "pMax" are NULL
- **VmbErrorWrongType:** The type of feature "name" is not Integer
- **VmbErrorNotFound:** If the feature was not found

VmbFeatureIntIncrementQuery()

Query the increment of an integer feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	VmbInt64_t*	pValue	Value of the increment to get.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command

- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Integer
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter: If "name" or "pValue" is NULL**

Float

VmbFeatureFloatGet()

Get the value of a float feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	double*	pValue	Value to get

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Float
- **VmbErrorBadParameter:** If "name" or "pValue" is NULL
- **VmbErrorNotFound:** The feature was not found

VmbFeatureFloatSet()

Set the value of a float feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
in	double	value	Value to set

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Float
- **VmbErrorInvalidValue:** If "value" is not within valid bounds
- **VmbErrorNotFound:** The feature was not found

- **VmbErrorBadParameter:** If "name" is NULL
- **VmbErrorInvalidCall:** If called from frame callback

VmbFeatureFloatRangeQuery()

Query the range of a float feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	double*	pMin	Minimum value to be returned. May be NULL.
out	double*	pMax	Maximum value to be returned. May be NULL.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Float
- **VmbErrorNotFound:** The feature was not found
- **VmbBadParameter:** If "name" is NULL or "pMin" and "pMax" are NULL

Only one of the values may be queried if the other parameter is set to NULL, but if both parameters are NULL, an error is returned.

VmbFeatureFloatIncrementQuery()

Query the increment of an float feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	VmbBool_t *	pHasIncrement	"true" if this float feature has an increment.
out	double*	pValue	Value of the increment to get.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Integer
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter: If "name" or "pValue" is NULL**

Enum

VmbFeatureEnumGet()

Get the value of an enumeration feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	const char**	pValue	The current enumeration value. The returned value is a reference to the API value

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Enumeration
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "name" or "pValue" is NULL

VmbFeatureEnumSet()

Set the value of an enumeration feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
in	const char*	value	Value to set

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Enumeration

- **VmbErrorInvalidValue:** If "value" is not within valid bounds
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "name" ore "value" is NULL
- **VmbErrorInvalidCall:** If called from frame callback

VmbFeatureEnumRangeQuery()

Query the value range of an enumeration feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
out	const char**	pNameArray	An array of enumeration value names; may be NULL if pNumFilled is used for size query
in	VmbUint32_t	arrayLength	Number of elements in the array
out	VmbUint32_t *	pNumFilled	Number of filled elements; may be NULL if pNameArray is not NULL

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorMoreData:** The given array length was insufficient to hold all available entries
- **VmbErrorWrongType:** The type of feature "name" is not Enumeration
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "name" is NULL or "pNameArray" and "pNumFilled" are NULL

VmbFeatureEnumIsAvailable()

Check if a certain value of an enumeration is available.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
in	const char*	value	Value to check
out	VmbBool_t *	pIsAvailable	Indicates if the given enumeration value is available

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Enumeration
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "name" or "value" or "pIsAvailable" is NULL

VmbFeatureEnumAsInt()

Get the integer value for a given enumeration string value.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
in	const char*	value	The enumeration value to get the integer value for
out	VmbInt64_t*	pIntVal	The integer value for this enumeration entry

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Enumeration
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "name" or "value" or "pIntVal" is NULL

Converts a name of an enum member into an int value ("Mono12Packed" to 0x10C0006)

VmbFeatureEnumAsString()

Get the enumeration string value for a given integer value.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the feature
in	VmbInt64_t	intValue	The numeric value
out	const char**	pStringValue	The string value for the numeric value

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Enumeration
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "name" or "pStringValue" is NULL

Converts an int value to a name of an enum member (e.g. 0x10C0006 to "Mono12Packed")

VmbFeatureEnumEntryGet()

Get infos about an entry of an enumeration feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	featureName	Name of the feature
in	const char*	entryName	Name of the enum entry of that feature
out	VmbFeatureEnumEntry_t*	pFeatureEnumEntry	Infos about that entry returned by the API
in	VmbUint32_t	sizeofFeatureEnumEntry	Size of the structure

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorStructSize Size of VmbFeatureEnumEntry_t is not compatible with the API version**
- **VmbErrorWrongType:** The type of feature "name" is not Enumeration
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorBadParameter:** If "featureName" or "entryName" or "pFeatureEnumEntry" is NULL

String

VmbFeatureStringGet()

Get the value of a string feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the string feature
out	char*	buffer	String buffer to fill. May be NULL if pSizeFilled is used for size query.
in	VmbUint32_t	bufferSize	Size of the input buffer
out	VmbUint32_t*	pSizeFilled	Size actually filled. May be NULL if buffer is not NULL.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorMoreData:** The given buffer size was too small
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorWrongType:** The type of feature "name" is not String

This function is usually called twice: once with an empty buffer to query the length of the string, and then again with a buffer of the correct length.

VmbFeatureStringSet()

Set the value of a string feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the string feature
in	const char*	value	Value to set

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorNotFound:** The feature was not found
- **VmbErrorWrongType:** The type of feature "name" is not String
- **VmbErrorInvalidValue:** If length of "value" exceeded the maximum length
- **VmbErrorBadParameter:** If "name" or "value" is NULL
- **VmbErrorInvalidCall:** If called from frame callback

VmbFeatureStringMaxlengthQuery()

Get the maximum length of a string feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the string feature
out	VmbUint32_t*	pMaxLength	Maximum length of this string feature

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not String
- **VmbErrorBadParameter:** If "name" or "pMaxLength" is NULL

Boolean

VmbFeatureBoolGet()

Get the value of a boolean feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the boolean feature
out	VmbBool_t *	pValue	Value to be read

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Boolean
- **VmbErrorNotFound:** If feature is not found
- **VmbErrorBadParameter:** If "name" or "pValue" is NULL

VmbFeatureBoolSet()

Set the value of a boolean feature.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the boolean feature
in	VmbBool_t	value	Value to write

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Boolean
- **VmbErrorInvalidValue:** If "value" is not within valid bounds
- **VmbErrorNotFound:** If the feature is not found

- **VmbErrorBadParameter:** If "name" is NULL
- **VmbErrorInvalidCall:** If called from frame callback

Command

VmbFeatureCommandRun()

Run a feature command.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the command feature

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Command
- **VmbErrorNotFound:** Feature was not found
- **VmbErrorBadParameter:** If "name" is NULL

VmbFeatureCommandIsDone()

Check if a feature command is done.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the command feature
out	VmbBool_t *	pIsDone	State of the command.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Command
- **VmbErrorNotFound:** Feature was not found
- **VmbErrorBadParameter:** If "name" or "pIsDone" is NULL

Raw

VmbFeatureRawGet()

Read the memory contents of an area given by a feature name.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the raw feature
out	char*	pBuffer	Buffer to fill
in	VmbUint32_t	bufferSize	Size of the buffer to be filled
out	VmbUint32_t*	pSizeFilled	Number of bytes actually filled

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Register
- **VmbErrorNotFound:** Feature was not found
- **VmbErrorBadParameter:** If "name" or "pBuffer" or "pSizeFilled" is NULL

This feature type corresponds to a top-level "Register" feature in GenICam. Data transfer is split up by the transport layer if the feature length is too large. You can get the size of the memory area addressed by the feature "name" by VmbFeatureRawLengthQuery().

VmbFeatureRawSet()

Write to a memory area given by a feature name.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the raw feature
in	const char*	pBuffer	Data buffer to use
in	VmbUint32_t	bufferSize	Size of the buffer

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Register
- **VmbErrorNotFound:** Feature was not found
- **VmbErrorBadParameter:** If "name" or "pBuffer" is NULL
- **VmbErrorInvalidCall:** If called from frame callback

This feature type corresponds to a first-level "Register" node in the XML file. Data transfer is split up by the transport layer if the feature length is too large. You can get the size of the memory area addressed by the feature "name" by VmbFeatureRawLengthQuery().

VmbFeatureRawLengthQuery()

Get the length of a raw feature for memory transfers.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that exposes features
in	const char*	name	Name of the raw feature
out	VmbUint32_t*	pLength	Length of the raw feature area (in bytes)

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorWrongType:** The type of feature "name" is not Register
- **VmbErrorNotFound:** Feature not found

- **VmbErrorBadParameter:** If "name" or "pLength" is NULL

This feature type corresponds to a first-level "Register" node in the XML file.

Feature invalidation

VmbFeatureInvalidationRegister()

Register a VmbInvalidationCallback callback for feature invalidation signaling.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that emits events
in	const char*	name	Name of the event
in	VmbInvalidationCallback	callback	Callback to be run, when invalidation occurs
in	void*	pUserContext	User context passed to function

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode

Any feature change, either of its value or of its access state, may be tracked by registering an invalidation callback. Registering multiple callbacks for one feature invalidation event is possible because only the combination of handle, name, and callback is used as key. If the same combination of handle, name, and callback is registered a second time, it overwrites the previous one.

VmbFeatureInvalidationUnregister()

Unregister a previously registered feature invalidation callback.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that emits events
in	const char*	name	Name of the event
in	VmbInvalidationCallback	callback	Callback to be removed

- **VmbErrorSuccess:** If no error

- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode

Since multiple callbacks may be registered for a feature invalidation event, a combination of handle, name, and callback is needed for unregistering, too.

Image preparation and acquisition

VmbFrameAnnounce()

Announce frames to the API that may be queued for frame capturing later.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle for a camera
in	const VmbFrame_t*	pFrame	Frame buffer to announce
in	VmbUint32_t	sizeofFrame	Size of the frame structure

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given camera handle is not valid
- **VmbErrorBadParameter:** The given frame pointer is not valid or "sizeofFrame" is 0
- **VmbErrorStructSize:** The given struct size is not valid for this version of the API

Allows some preparation for frames like DMA preparation depending on the transport layer. The order in which the frames are announced is not taken into consideration by the API. The method can be used to annouce a previously allocated frame buffer to the transport layer. Alternatively, in case "pFrame->buffer" points to NULL, the method will allocate and announce a new buffer. In this case "pFrame->buffer" contains the allocated buffer address on return.

VmbFrameRevoke()

Revoke a frame from the API.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle for a camera
in	const VmbFrame_t*	pFrame	Frame buffer to be removed from the list of announced frames

- **VmbErrorSuccess:** If no error

- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given camera handle is not valid
- **VmbErrorBadParameter:** The given frame pointer is not valid
- **VmbErrorStructSize:** The given struct size is not valid for this version of the API

The referenced frame is removed from the pool of frames for capturing images.

VmbFrameRevokeAll()

Revoke all frames assigned to a certain camera.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle for a camera

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given camera handle is not valid

VmbCaptureStart()

Prepare the API for incoming frames.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle for a camera

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorDeviceNotOpen:** Camera was not opened for usage
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode

VmbCaptureEnd()

Stop the API from being able to receive frames.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle for a camera

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid

Consequences of VmbCaptureEnd(): - The frame callback will not be called anymore

VmbCaptureFrameQueue()

Queue frames that may be filled during frame capturing.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle of the camera
in	const VmbFrame_t*	pFrame	Pointer to an already announced frame
in	VmbFrameCallback	callback	Callback to be run when the frame is complete. NULL is Ok.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given frame is not valid
- **VmbErrorStructSize:** The given struct size is not valid for this version of the API

The given frame is put into a queue that will be filled sequentially. The order in which the frames are filled is determined by the order in which they are queued. If the frame was announced with VmbFrameAnnounce() before, the application has to ensure that the frame is also revoked by calling VmbFrameRevoke() or VmbFrameRevokeAll() when cleaning up.

VmbCaptureFrameWait()

Wait for a queued frame to be filled (or dequeued).

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle of the camera
in	const VmbFrame_t*	pFrame	Pointer to an already announced & queued frame
in	VmbUint32_t	timeout	Timeout (in milliseconds)

- **VmbErrorSuccess:** If no error
- **VmbErrorTimeout:** Call timed out
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid

VmbCaptureQueueFlush()

Flush the capture queue.

	Type	Name	Description
in	const VmbHandle_t	cameraHandle	Handle of the camera to flush

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid

Control of all the currently queued frames will be returned to the user, leaving no frames in the capture queue. After this call, no frame notification will occur until frames are queued again.

Interface Enumeration & Information

VmbInterfacesList()

List all the interfaces currently visible to VimbaC.

	Type	Name	Description
out	VmbInterfaceInfo_t*	pInterfaceInfo	Array of VmbInterfaceInfo_t, allocated by the caller. The interface list is copied here. May be NULL.
in	VmbUint32_t	listLength	Number of entries in the caller's pList array
out	VmbUint32_t*	pNumFound	Number of interfaces found (may be more than listLength!) returned here.
in	VmbUint32_t	sizeofInterfaceInfo	Size of one VmbInterfaceInfo_t entry

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorStructSize:** The given struct size is not valid for this API version
- **VmbErrorMoreData:** The given list length was insufficient to hold all available entries
- **VmbErrorBadParameter:** If "pNumFound" was NULL

All the interfaces known via GenICam TransportLayers are listed by this command and filled into the provided array. Interfaces may correspond to adapter cards or frame grabber cards or, in the case of FireWire to the whole 1394 infrastructure, for instance. This function is usually called twice: once with an empty array to query the length of the list, and then again with an array of the correct length.

VmbInterfaceOpen()

Open an interface handle for feature access.

	Type	Name	Description
in	const char*	idString	The ID of the interface to get the handle for (returned by VmbInterfacesList())
out	VmbHandle_t*	pInterfaceHandle	The handle for this interface.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorNotFound:** The designated interface cannot be found
- **VmbErrorBadParameter:** If "pInterfaceHandle" was NULL

An interface can be opened if interface-specific control or information is required, e.g. the number of devices attached to a specific interface. Access is then possible via feature access methods.

VmbInterfaceClose()

Close an interface.

	Type	Name	Description
in	const VmbHandle_t	interfaceHandle	The handle of the interface to close.

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid

After configuration of the interface, close it by calling this function.

Ancillary data

VmbAncillaryDataOpen()

Get a working handle to allow access to the elements of the ancillary data via feature access.

	Type	Name	Description
in	VmbFrame_t*	pFrame	Pointer to a filled frame
out	VmbHandle_t*	pAncillaryDataHandle	Handle to the ancillary data inside the frame

- **VmbErrorSuccess:** No error
- **VmbErrorBadHandle:** Chunk mode of the camera was not activated. See feature ChunkModeActive
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command

This function can only succeed if the given frame has been filled by the API.

VmbAncillaryDataClose()

Destroy the working handle to the ancillary data inside a frame.

	Type	Name	Description
in	VmbHandle_t	ancillaryDataHandle	Handle to ancillary frame data

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid

After reading the ancillary data and before re-queuing the frame, ancillary data must be closed.

Memory/Register access

VmbMemoryRead()

Read an array of bytes.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that allows memory access
in	VmbUint64_t	address	Address to be used for this read operation
in	VmbUint32_t	bufferSize	Size of the data buffer to read
out	char*	dataBuffer	Buffer to be filled
out	VmbUint32_t*	pSizeComplete	Size of the data actually read

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode

VmbMemoryWrite()

Write an array of bytes.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that allows memory access
in	VmbUint64_t	address	Address to be used for this read operation
in	VmbUint32_t	bufferSize	Size of the data buffer to write
in	const char*	dataBuffer	Data to write
out	VmbUint32_t*	pSizeComplete	Number of bytes successfully written; if an error occurs this is less than bufferSize

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command

- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorMoreData:** Not all data were written; see pSizeComplete value for the number of bytes written

VmbRegistersRead()

Read an array of registers.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that allows register access
in	VmbUint32_t	readCount	Number of registers to be read
in	const VmbUint64_t*	pAddressArray	Array of addresses to be used for this read operation
out	VmbUint64_t*	pDataArray	Array of registers to be used for this read operation
out	VmbUint32_t*	pNumCompleteReads	Number of reads completed

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorIncomplete:** Not all the requested reads could be completed

Two arrays of data must be provided: an array of register addresses and one for corresponding values to be read. The registers are read consecutively until an error occurs or all registers are written successfully.

VmbRegistersWrite()

Write an array of registers.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that allows register access
in	VmbUint32_t	writeCount	Number of registers to be written
in	const VmbUint64_t*	pAddressArray	Array of addresses to be used for this write operation
in	const VmbUint64_t*	pDataArray	Array of reads to be used for this write operation
out	VmbUint32_t*	pNumCompleteWrites	Number of writes completed

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorIncomplete:** Not all the requested writes could be completed

Two arrays of data must be provided: an array of register addresses and one with the corresponding values to be written to these addresses. The registers are written consecutively until an error occurs or all registers are written successfully.

VmbCameraSettingsSave()

Saves all feature values to XML file.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that allows register access
in	const char*	fileName	Name of XML file to save settings
in	VmbFeaturePersistSettings_t*	pSettings	Settings struct
in	VmbUint32_t	sizeofSettings	Size of settings struct

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command

- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorBadParameter:** If "fileName" is NULL

Camera must be opened beforehand and function needs corresponding handle. With given filename parameter path and name of XML file can be determined. Additionally behaviour of function can be set with providing 'persistent struct'.

VmbCameraSettingsLoad()

Load all feature values from XML file to device.

	Type	Name	Description
in	const VmbHandle_t	handle	Handle for an entity that allows register access
in	const char*	fileName	Name of XML file to save settings
in	VmbFeaturePersistSettings_t*	pSettings	Settings struct
in	VmbUint32_t	sizeofSettings	Size of settings struct

- **VmbErrorSuccess:** If no error
- **VmbErrorApiNotStarted:** VmbStartup() was not called before the current command
- **VmbErrorBadHandle:** The given handle is not valid
- **VmbErrorInvalidAccess:** Operation is invalid with the current access mode
- **VmbErrorBadParameter:** If "fileName" is NULL

Camera must be opened beforehand and function needs corresponding handle. With given filename parameter path and name of XML file can be determined. Additionally behaviour of function can be set with providing 'settings struct'.