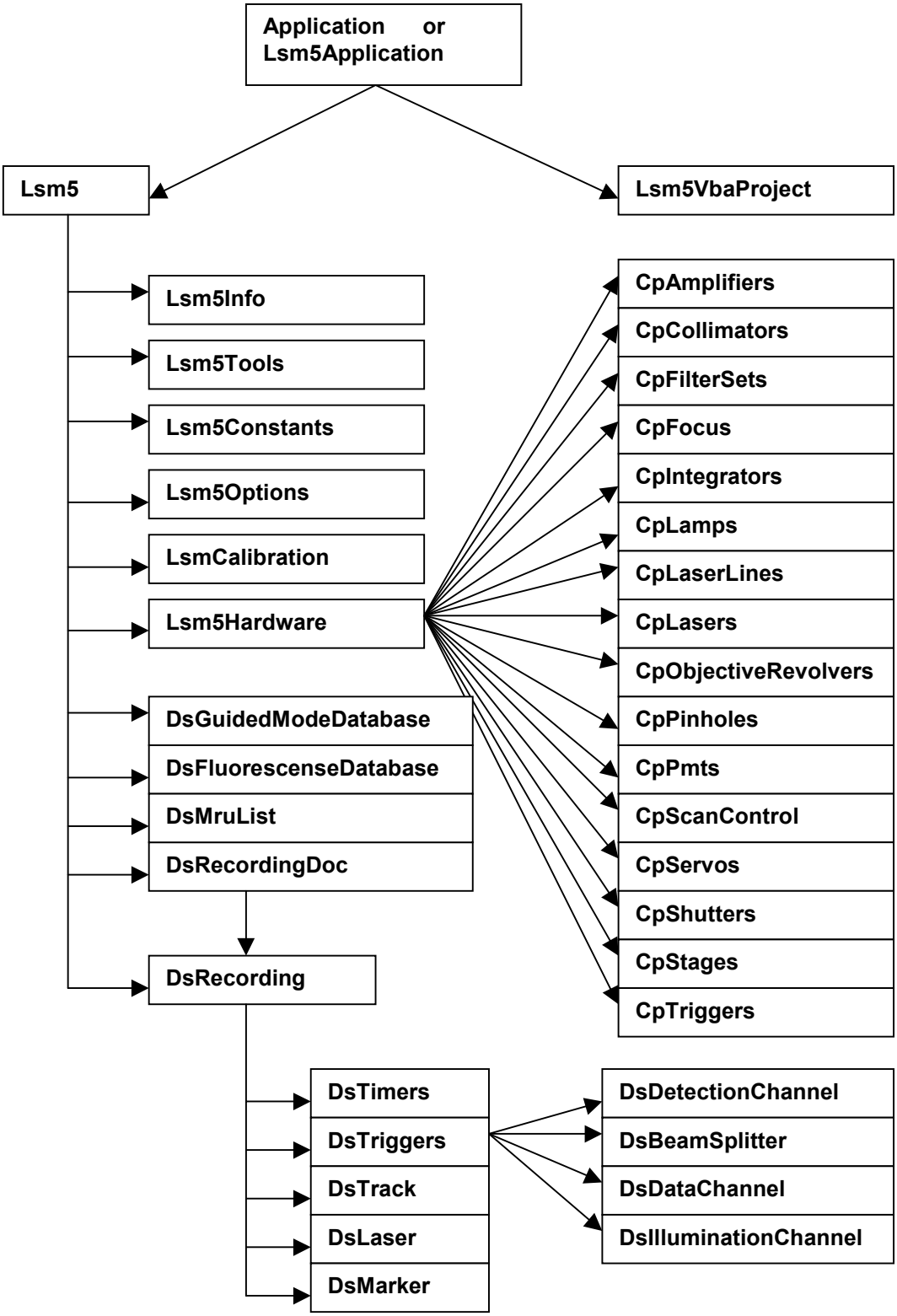
SW-Documentation for LSM 5 Macro interface Version 3.2

1. Introduction

The LSM510 Interface server (LSM5Vba.dll) is accessible via an OLE-automation interface. The figure in capture 3 shows the call tree.
Connecting to the LSM 510 Interface server yields the dispatch interface to the root object "Lsm5Application". Sub objects are accessible with the listed methods yielding a dispatch pointer to the sub objects.

2. Table of Contents

3. The Interface hierarchy overview.



4. Interface descriptions

4.1. The root entry interface; Lsm5Application, Application

Lsm5Application, Application	
"Lsm5Application" is the root interface of an external connected program. "Application" is the root interface used in all macros.	
Properties	
BSTR **ProjectFilePath**	Last used macro project path, path where the macros stored
Lsm5 * **Lsm5**	Access to the LSM entry interface. Lsm5 specific functions
IDispatch* **Fcs**	Access to the FCS entry object.
Boolean **MacroRunning**	Gets if any macro is running.
Methods	
Boolean **ShowVBAEditor** (Boolean Show)	Displays or hides the VBA editor.
Lsm5VbaProject * **ProjectLoad** (BSTR ProjectFileName, short* Success)	Load the specified macro project file
Lsm5VbaProject * **ProjectLoadDialog** (short* Success)	Show a file dialog to select a macro file an load the specified macro project file
Void **ProjectUnLoad** (BSTR Title)	Unload the specified macro project file. If the macro is running, the execution will be stopped.
Long **ProjectCount** ()	Gets the number of loaded macro projects
Lsm5VbaProject* **ProjectNew** (short* Success)	Creates a new macro project
BSTR **ProjectTitle** (long Index, short* Success)	Get the title of a loaded macro project file 'Index' (IN) index of the loaded macro project list 'Success' (OUT) return value. Zero means FALSE.
BSTR **ProjectPath** (long Index, short* Success)	Get the full path of a loaded project 'Index' (IN) index of the loaded macro project list 'Success' (OUT) return value. Zero means FALSE.
Boolean **MacroRecordStart** (BSTR Title, BSTR MacroName, BSTR Description)	Starts the macro recording. Parameters can be blank.
Void **MacroRecordEnd** (Boolean EditNow)	End the macro recording with | without starts the VBA editor.
Void **MacroRecordCancel**()	Cancels the macro recording

Boolean **MacroRun** (BSTR ProjectPath, BSTR MacroName)	Loads the macro project file and runs the specified macro
Boolean **MacroRunByTitle** (BSTR Title, BSTR Macro)	Runs a macro from the loaded project title
Lsm5VbaProject* **ProjectObjectByIndex** (long Index, short* Success)	Gets the object of a loaded macro project by index.
Boolean **ProjectCloseAll** ()	Closes all loaded macro project files.
Void **SetParentWindowHandle** (OLE_HANDLE ParentWindow)	Gives the server the parent window to correct the z-order.
Lsm5VbaProject * **GetProjectObject** (BSTR Title, short* Success)	Returns the interface to the given project title.

4.2. Lsm5

Lsm5	
"Lsm5" is the root interface of all LSM5 specific functions and properties.	
Properties	
Double **SelectedUpperPosition**	"SelectedUpperPosition" can be used for definition of the scan range and focus position z-scan and stack scan using the mouse in the image window. The procedure is: 1. Call "ScanVerticalPlaneAndSelectRange" 2. After the user has changed the position of one of the three lines on the screen with the mouse a event is fired (see "SetEventHandler"). 3. The event handler can call "SelectedUpperPosition", " SelectedLowerPosition" and "SelectedCurrentPosition" to obtain the new positions of the lines. Returns - the z-position of the upper range selection line in microns relative to the lower frame of the displayed image.
Double **SelectedLowerPosition**	"SelectedLowerPosition" can be used for definition of the scan range and focus position z-scan and stack scan using the mouse in the image window. See "SelectedUpperPosition" for details. Returns - the z-position of the lower range selection line in microns relative to the lower frame of the displayed image.
double **SelectedCurrentPosition**	"SelectedCurrentPosition" can be used for definition of the scan range and focus position z-scan and stack scan using the mouse in the image window. See "SelectedUpperPosition" for details. Returns - the z-position of the focus selection line in microns relative to the lower frame of the displayed image.
DsRecordingDoc* **DsRecordingActiveDocObject**	"DsRecordingActiveDocObject" returns a dispatch pointer to a "DsRecordingDoc" interface of the most recently used recording (image) document. The most recently used changes when the user clicks in to an image window or modifies pixel data by calculations or scan operations or image load operations.
DsMruList* **MruDatabases**	"MruDatabases" returns a dispatch pointer to a "DsMruList" interface which is a collection of database names of the databases that where most recently used. The collection contains up to 5 database names.
DsGuidedModeDatabase* **GuidedModeDatabase**	Get the Guided mode database object.
Lsm5Constants* **Constants**	Returns a dispatch pointer to the program constant object.
DsRecordingDoc* **DsRecordingDocObject** (long Index, short* Success)	Get the recording document dispatch pointer at the specified index.

Methods	
Lsm5Tools* **Tools** ()	Returns a dispatch pointer to the Tools Object.
Boolean **Login** (boolean BootDummy)	"Login" is part of the process to connect the "InterfaceServer" with the "ControlProgram". "Login" creates the "DsRecording" object holding the scan parameters if not already exists. On startup one should call "Login" first and then "BootHardware" to connect the "InterfaceServer" to the "ControlProgram". "Login" returns a nonzero value if the "DsRecording" object was successfully created. "Login" returns zero in case of insufficient memory or when no dongle could be found. BootDummy(IN) - If set to zero the LSM-Hardware is booted. If unequal to zero the "ControlProgram" is booted in "dummy mode" which means that the hardware is not accessible. Returns - unequal zero if successful.
Boolean **IsLoggedIn** ()	Check if logged in.
Lsm5Hardware* **Hardware** ()	Returns a dispatch pointer to the hardware objects.
DsRecording* **DsRecording** ()	"DsRecording" returns a dispatch pointer which contains scan parameters used for scan operations started with "StartScan" and "StartContinuousScan". "Login" must be called before.
Lsm5Info* **Info** ()	Returns a dispatch pointer to the info object.
DsRecordingDoc* **StartScan** ()	"StartScan" starts a single common scan operation which can be a "linescan", "framescan", "z-scan" or "stack-scan" depending on the scan mode specified in the "DsRecording" object accessible via "Recording". Special scan operations like auto B&C or range selection have there own start methods. After the scan operation has finished, an event is fired (see "SetEventHandler"). Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.

DsRecordingDoc* **StartContinuousScan** ()	"StartContinuousScan" starts a continuos common scan operation which can be a "linescan", "framescan", "z-scan" or "stack-scan" depending on the scan mode specified in the "DsRecording" object accessible via "Recording". For time series scan memory is provided only for one time index and the timing specified in "Recording" is used. After the scan operation has finished an event is fired (see "SetEventHandler"). Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.
DsRecordingDoc* **StartAutoBC** (long Multiplexorder)	"StartAutoBC" starts a automatic brightness and contrast adjustment scan operation. After the scan operation has finished an event is fired (see "SetEventHandler"). Multiplexorder(IN) : Determines for which track we will start the autobc process Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.
DsRecordingDoc* **NewScanWindow** ()	"NewScanWindow" creates a new empty (black) recording document and a dispatch pointer to the "DsRecordingDoc" interface of that new recording document. If the user does not activate an other scan window the next scan operations goes to this window.
DsRecordingDoc* **OpenRecording** (BSTR DataBase, long Index)	"OpenRecording" opens a recording document from file. First the database with the specified name is opened. Then the method looks for the specified index in the column "IdRecording" of the table "Recordings" in the database. If the row was found, the method looks for the entry in the column "SampleData" of the same row, which contains the image file path name. Then the image file is loaded. DataBase (IN) - name of the database which contains a name of the image file. Index (IN) - value in the "Irecording" column of the "Recordings" table in the database containing the data set for the image to load. returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document of the loaded image file or a null-pointer in case of an error.

Boolean **SaveRecording** ()	“SaveRecording” looks for the most recently used recording document. If one, the contents of the document are saved to database and image file. If the recording document was already save to a database the same record set and image file name as for the last save operation is used. If the recording document was not saved yet the “Save” dialog box appears on the screen. returns - unequal zero if there was no error.
Boolean **SaveRecordingAs** ()	“SaveRecordingAs” looks for the most recently used recording document. If one, the “Save As” dialog box is called where the user can choose the database and recording name. If the user presses “Ok” the contents of the recording document are saved to database and image file. returns - unequal zero if there was no error.

Boolean **OpenDatabase** (BSTR Name)	"OpenDatabase" opens a database document for the database with the specified path name. Name (IN) - Full path name of the database to open returns - unequal zero if the database was successfully opened
Boolean **NewDatabase** (BSTR Path)	"NewDatabase" creates a new database with the specified path name and opens the database document for the database. Path (IN) - Full path name of the database to create returns - unequal zero if the database was successfully created and opened
Boolean **Logout** ()	"Logout" closes all image documents and all database documents. Then the "DataServer" is disconnected from the "ControlProgram". If "Logout" returns zero
void **OptionsDialog** ()	"OptionsDialog" calls the "Options" dialog for choosing user specific program-global settings.
void **FullScreen** ()	"FullScreen" looks for the most recently used recording document. If one and if possible, the contents of the image window are displayed in a new created pop-up window which fills the whole screen (first monitor in case of multi monitor configuration). after a mouse click or a keystroke the pop-up window vanishes.
void **StopScan** ()	Stops the current scan.
Lsm5Options* **Options** ()	"Options" returns a dispatch pointer to a "Lsm5Options" interface with user specific program-global settings.
Long **RecordingDocCount** ()	returns the number of recording (image) documents that are open in the current session.
Boolean **Reboot** ()	Reboot the hardware. Not supported in this version.
Boolean **ExportRecording** ()	"ExportRecording" looks for the most recently used recording document. If one, the "Export" dialog box for file export is called.
DsRecordingDoc* **ImportRecording** (BSTR Path)	"ImportRecording" opens the specified image file. If "Path" is an empty string the "Import" file select box is called. Path (IN) - name of the image file to import. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document of the loaded image file or a null-pointer in case of an error.

Boolean **ExportToGuidedMode** ()	"ExportToGuidedMode" looks for the most recently used recording document. If one, the scan parameters of this recording document are used to create a new record set in the guided mode database. In "GuidedModeDatabase" is described what happens if the guided mode database does not exist. returns - unequal zero if the record set was successfully created.
Boolean **CloseAllImageWindows** (boolean AskForSaveModified)	"CloseAllImageWindows" closes all image windows and destroy the connected recording documents. "CloseAllImageWindows" also removes all modeless calculation dialogs. AskForSaveModified (IN) - specifies whether the user should be requested to save unmodified documents or not. Note that if the user option "Save Prompt at closing modified windows" is off the user is not requested anyway. returns - zero if the user has pressed the "Cancel" button in one save-request message box and unequal zero if not.
DsRecordingDoc* **ScanFrameAndSelectHorizontal** (double InitialLinePositionX, double InitialLinePositionY)	"ScanFrameAndSelectHorizontal" scans one plane as specified by the parameters in the "DsRecording" object accessible via "Recording" (The scan mode there is ignored). After the scan operation has finished an event is fired (see "SetEventHandler"). Then a horizontal line is displayed in the scan window. The user can move the line to specify a position for a z-Scan or Range select scan. An event is fired after the user has moved the line (see "SetEventHandler"). InitialLinePosition (IN) - specifies the initial position for the displayed line in microns relative to the center of the scan field. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.

DsRecordingDoc* **ScanVerticalFrameAndSelectRange** (double InitialUpperPosition, double InitialLowerPosition, double InitialActualPosition)	"ScanVerticalFrameAndSelectRange" scans one vertical plane (z-Scan). After the scan operation has finished an event is fired (see "SetEventHandler"). Most of the parameters in the "DsRecording" object accessible via "Recording" are used. The z-Range "DsRecording" in pixels and microns is doubled for the scan operation. After the scan operation three a horizontal lines are displayed in the scan window. Two red lines for range selection and a green line for selection of the current position. The user can move the lines to specify a new range and focus position for a z-scan or stack-scan. An event is fired after the user has moved one of the lines (see "SetEventHandler"). InitialUpperPosition (IN) - specifies the initial position for the displayed line of the upper range limit in microns relative to the lower frame of the scanned image. InitialLowerPosition (IN) - specifies the initial position for the displayed line of the lower range limit in microns relative to the lower frame of the scanned image. InitialActualPosition (IN) - specifies the initial position for the displayed line of the actual focus position in microns relative to the lower frame of the scanned image. returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.
Boolean **CalculateProjection** ()	"CalculateProjection" looks for the most recently used recording document. If one, the modal "Projection" dialog box is called and if the user has pressed the "Ok" button the projection calculation is started.
Boolean **CalculateDepthCoding** ()	"CalculateDepthCoding" looks for the most recently used recording document. If one, the modal "Depth Coding" dialog box is called and if the user has pressed the "Ok" button the depth coding calculation is started.
Boolean **CalculateStereoImage** ()	"CalculateStereoImage" looks for the most recently used recording document. If one, the modal "Stereo Images" dialog box is called and if the user has pressed the "Ok" button the stereo images calculation is started.
Boolean **CalculateRatio** ()	"CalculateRatio" looks for the most recently used recording document. If one, the modal "Ratio" dialog box is called and if the user has pressed the "Ok" button the off-line ratio calculation is started.

Boolean **CalculateAdd** ()	“CalculateAdd” looks for the most recently used recording document. If one, the modal “Add” dialog box is called and if the user has pressed the “Ok” button the summation is started.
Boolean **CalculateSubtract** ()	“CalculateSubtract” looks for the most recently used recording document. If one, the modal “Subtract” dialog box is called and if the user has pressed the “Ok” button the calculation is started.

Boolean **CalculateMultiply** ()	"CalculateMultiply" looks for the most recently used recording document. If one, the modal "Multiply" dialog box is called and if the user has pressed the "Ok" button the calculation is started.
Boolean **CalculateDivide** ()	"CalculateDivide" looks for the most recently used recording document. If one, the modal "Divide" dialog box is called and if the user has pressed the "Ok" button the calculation is started.
Boolean **CalculateFilter** ()	"CalculateFilter" looks for the most recently used recording document. If one, the modal "Filter" dialog box is called and if the user has pressed the "Ok" button the filter operation is started.
Boolean **InterpolateContrast** ()	"InterpolateContrastModlessDialog" calls the modal "Interpolate Brightness and Contrast" dialog. returns - unequal zero if the dialog was successfully created
Boolean **DisplayChannelCopy** ()	"DisplayChannelCopyModlessDialog" calls the modal "Copy" dialog. returns - unequal zero if the dialog was successfully created
Boolean **TestGrid** ()	"DisplayChannelCopy" looks for the most recently used recording document. If one, and the actual display window of the document is in "XY" display mode a test grid with 20 pixels line distance and line length of 512 pixels is generated in the vector overlay of the display. Three cycles are also generated in the vector overlay. returns - unequal zero if most recently used recording exist.
DsRecordingDoc* **DuplicateDsRecordingDocObject** ()	"DuplicateDsRecordingDocObject" looks for the most recently used recording document. If one, a new recording document is created and the scan memory of the source recording document is to the new recording document. returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created or a null-pointer if there is no source recording document.
DsRecordingDoc* **HorizontalSelectLine** (double InitialLinePositionX, double InitialLinePositionY)	Same as "ScanFrameAndSelectHorizontal" except that the xy-image is only scanned if the most recently used scan window does not already contain a x-y-image. "HorizontalSelectLine" can be called instead of "ScanFrameAndSelectHorizontal" to avoid unnecessary bleaching. InitialLinePosition (IN) - specifies the initial position for the displayed line in microns relative to the center of the scan field. returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.

DsRecording* **CreateBackupRecording** ()	"CreateBackupRecording" creates a new "DsRecording" object which can be used to remember the whole set of scan parameters and can be used to temporary modify scan parameters and then restore the old state of the scan parameters later. "DsRecording" has a method with the name "Copy" for this purpose. Note that the returned dispatch pointer is valid until the "Release" member of the "IUnknown" interface is called. This will happen if the object runs out of scope in Visual Basic. returns - a dispatch pointer to the "DsRecording" Interface of the new created "DsRecording" object.
DsRecordingDoc* **StartInvisibleScan** ()	same as "StartScan" except that a new recording document is created in every case but no image window is created for that recording document. returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.
DsRecordingDoc* **StartInvisibleContinuousScan** ()	same as "StartContinuousScan" except that a new recording document is created in every case but no image window is created for that recording document. returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document where the scan data goes in or a null-pointer in case of an error.
Boolean **ReuseFromGuidedMode** (long IdRecording)	"ReuseFromGuidedMode" opens the guided mode database if not already open moves to the specified record set in the "Recordings" table, reads the record set, copies the scan parameter to the "DsRecording" object accessible by "Recording" and fires a "Reuse" event. That means these scan parameters are now the actual scan parameters. See "SetEventHandler" for details of the "Reuse" event. returns - unequal zero if successful
DsFluorescenseDatabase* **OpenFluorescenceDatabase** (BSTR DatabasePath, BSTR SqlString)	"OpenFluorescenceDatabase" returns a dispatch pointer to a "DsFluorescenseDatabase" interface which is a collection of objects containing dye emission and excitation wavelengths. If the database does not exist it is created in the specified path by loading it from the resource part of the "dll". If "OpenFluorescenceDatabase" was already successfully called the database closed and reopened. DatabasePath (IN) - Path name of the fluorescence database SqlString (IN) - SQL string to open the record set in the database. If all record sets are request use "select * from [Fluorescence]". returns - a dispatch pointer to the "DsFluorescenseDatabase" interface if successful and a null-pointer if not.

DsChannelColors* **ChannelColors** ()	“ChannelColors” returns a dispatch pointer to a “DsChannelColors” interface which is a collection of colors used for the selection of channel colors in the image display.
boolean **DefineRoisModlessDialog** ()	“DefineRoiModlessDialog” calls a modless dialog with a ROI editor where the user can define the ROIs for scan operations. If there is a ROI editor dialog for bleach ROIs visible then this editor dialog is replaced. returns - TRUE , if the dialog is now visible else “FALSE”.
boolean **GetIsDummyBooted** ()	Determines if the system was booted as dummy or not.
boolean **ShowStatusDialog** (boolean Visible)	“ShowStatusDialog” shows or hides the “scan information” display window. Visible (IN) - if “TUE” the dialog willl be shown and if “FASLE” the dialog will be hidden. returns - “TRUE “ if successful
Boolean I**sStatusDialogVisible** ()	“IsStatusDialogVisible” can be used to obtain the information whether the “scan information” display window is visible or not. returns - “TRUE “ if the “scan information” display window is visible and “FALSE” if not.
Boolean **SetMarker** (long MarkerNumber)	“SetMarker” records the actual time and the description string of the specified marker for the display of experimental annotions. NumberMarker (IN) - zero based index of the “DsMarker” object of the scan parameters accessible via “Recording”. returns - “TRUE “ if successful
Boolean **Bleach** (long MultiplexOrder)	“Bleach” starts a bleach operation. MultiplexOrder (IN) - one based multiplex order of the bleach track in the “DsTrack” collection of the scan parameters accessible via “Recording”. returns - “TRUE “ if bleach operation was successfully started.

Boolean **DefineBleachRoisModlessDialog** ()	"DefineRoiModlessDialog" calls a modless dialog with a ROI editor where the user can define the ROIs for the specified bleach operation. If there is already a ROI editor dialog for bleach ROIs or scan ROIs visible then this editor dialog is replaced. MultiplexOrder (IN) - one based multiplex order of the bleach track in the "DsTrack" collection of the scan paramaeters acessible via "Recording". Returns - TRUE if the dialog is now visible else "FALSE".
Boolean **ShowLaserDialog** ()	Shows the laser control dialog.
DsRecordingDoc* **StartScanMeanOfRois** ()	Starts a "Mean of ROI" scan.
DsRecordingDoc* **NewRecordingWithCurrentBandC** ()	"NewRecordingWithCurrentBandC" looks for the most recently used recording document. If one, a new recording document is created and the scan memory of the source recording document is translated and stored to the new recording document. The translation is done by a LUT containing the brightness and contrast properties adjusted by the user in the source recording document. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created or a null-pointer if there is no source recording document.
Boolean **GetIsTimeSeriesPause** ()	Checks if a time series scan is in the pause mode.
Void **PauseTimeSeries** ()	Pauses a time series scan.
void **ResumeTimeSeries** (double NewIntervalTime)	Resume a time series scan after a pause.
Boolean **CalculatePixelShiftDialog** ()	Displays the Pixel shift dialog.
Boolean **IsValidRoi** ()	Checks if the ROI is valid.
Boolean I**sValidBleachRoi** ()	Checks if the bleach ROI is valid.
DsRecordingDoc* **StartFastScan** ()	Starts a fast scan. Scan speed 10 with a 512x 512 image.

DsRecordingDoc* **CalculateProjectionDlg** ()	"CalculateProjection" looks for the most recently used recording document. If one, the modal "Projection" dialog box is called and if the user has pressed, the "Ok" button the projection calculation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the projection operation or a null-pointer if there is no source-recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.
DsRecordingDoc* **CalculateDepthCodingDlg** ()	"CalculateDepthCoding" looks for the most recently used recording document. If one, the modal "Depth Coding" dialog box is called and if the user has pressed the "Ok" button the depth coding calculation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the depth coding operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.
DsRecordingDoc* **CalculateStereoImageDlg** ()	"CalculateStereoImage" looks for the most recently used recording document. If one, the modal "Stereo Images" dialog box is called and if the user has pressed the "Ok" button the stereo images calculation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the stereo images operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box
DsRecordingDoc* **CalculateRatioDlg** ()	"CalculateRatio" looks for the most recently used recording document. If one, the modal "Ratio" dialog box is called and if the user has pressed the "Ok" button the off-line ratio calculation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the off-line ratio operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.

DsRecordingDoc* **CalculateAddDlg** ()	"CalculateAdd" looks for the most recently used recording document. If one, the modal "Add" dialog box is called and if the user has pressed the "Ok" button the summation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.
DsRecordingDoc* **CalculateSubtractDlg** ()	"CalculateSubtract" looks for the most recently used recording document. If one, the modal "Subtract" dialog box is called and if the user has pressed the "Ok" button the calculation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.
DsRecordingDoc* **CalculateMultiplyDlg** ()	"CalculateMultiply" looks for the most recently used recording document. If one, the modal "Multiply" dialog box is called and if the user has pressed the "Ok" button the calculation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.
DsRecordingDoc* **CalculateDivideDlg** ()	"CalculateDivide" looks for the most recently used recording document. If one, the modal "Divide" dialog box is called and if the user has pressed the "Ok" button the calculation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.
DsRecordingDoc* **CalculateFilterDlg** ()	"CalculateFilter" looks for the most recently used recording document. If one, the modal "Filter" dialog box is called and if the user has pressed the "Ok" button the filter operation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the filter operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.

DsRecordingDoc* **DisplayChannelCopyDlg** ()	"DisplayChannelCopyDlg" looks for the most recently used recording document. If one, the modal "Copy" dialog box is called and if the user has pressed the "Ok" button the copy operation is started. Returns - a dispatch pointer to the "DsRecordingDoc" interface of the recording document that was new created by the copy operation or a null-pointer if there is no source recording document. A null-pointer is returned also if the user has pressed the "Cancel"-Button in the dialog box.
double **SelectedHorizontalLineX** ()	Gets the selected horizontal Line in X direction.
double **SelectedHorizontalLineY** ()	Gets the selected horizontal Line in Y direction.
DsRecordingDoc* **MakeNewImageDocument** (long DimensionX, long DimensionY, long DimensionZ, long DimensionT, long Channels, long DataType, short Visible)	Creates a new DsRecordingDocument(image window) with the given parameters Returns the dispatch pointer to DsRecordingDoc
DsRecordingDoc* **MakeNewLinescanImageDocument** (long DimensionX, long DimensionZ, long DimensionT, long Channels, long DataType, short Visible)	Creates a new DsRecordingDocument(image window) with the given parameters Returns the dispatch pointer to DsRecordingDoc
Boolean **CloseAllDatabaseWindows** ()	"CloseAllDatabaseWindows" closes all database windows. Returns - zero if the user has pressed the "Cancel" button in one save-request message box and unequal zero if not.
Void **CloseAllDialogs** ()	"CloseAllDialogs" closes all dialog windows. "CloseAllDialogs" also removes all modeless calculation dialogs.
DsRecordingDoc* **GetLastScanDestination** ()	Returns the dispatch pointer to the last used image document (DsRecordingDoc).
Boolean **OpenGuidedModeDatabase** ()	Displays the dialog to open the "GuidedModeDatabase" as a normal Database.
Boolean **InitDone** ()	Gets if the initialization of the interface dll is done.
Boolean **Initialize** ()	Initialize the interfaqce dll.

Boolean **Uninitialize** ()	Uninitialize the interfaqce dll. returns - “TRUE “ if successful
Boolean **MultiImagePrintWindow** ()	Shows the “multiimage print” dialog. returns - “TRUE “ if successful
Boolean **ShowStageControlDialog** (Boolean show)	Shows or hides the “Stage and Focus Control” dialog. show (IN) - if “TURE” the dialog willl be shown and if “FASLE” the dialog will be hidden. returns - “TRUE “ if successful
IDsGuidedModeDatabase* ppReturn **ImageDatabaseForReadAccess** (BSTR FileName)	Returns the dispatch pointer to DsGuidedModeDatabase interface for read access. show (IN) - Name of the database returns the dispatch pointer to the “DsGuidedModeDatabas” interface
IDsVectorOverlay* ppReturn **RoiList** ()	Returns the dispatch pointer to IDsVectorOverlay interface
Void **UpdateRoiDialog** ()	Updates the “Edit ROI” diaolg.
Boolean **ShowScannerCalibDialog** ()	Shows the “Scanner Calibration” dialog. returns - “TRUE “ if successful
Boolean **CameraDlg** ()	Shows the “camera” dialog. returns - “TRUE “ if successful
Boolean **DoPasswordCheckDialog** ()	Shows the “Enter Password” dialog and checkes the service password. returns - “TRUE “ if successful and the password is valid
Boolean **AutoDdsCorrection** ()	Calibrates the phase shift between the forward and backward movement of the scanner for bidirectional scan. returns - “TRUE “ if successful
Boolean **ShowSpectraDialog** (Boolean show)	Shows or hides the “spectra” dialog. show (IN) - if “TURE” the dialog willl be shown and if “FASLE” the dialog will be hidden. returns - “TRUE “ if successful
Void **SetSpectraDialogDisplayTrack** (long tracknr)	Set the track,that is displayed by “spectra” dialog. tracknr - number of the track
Boolean **ShowAOTFSelectDialog** (Boolean show)	Shows or hides the “Wavelenght Switch Control” dialog. show (IN) - if “TURE” the dialog willl be shown and if “FASLE” the dialog will be hidden. returns - “TRUE “ if successful
Boolean **CanShowAOTFSelectDialog** ()	Get if the the “Wavelenght Switch Control” dialog can be shown.

Boolean **CalculateDeconvolution** ()	Shows the devonvolution calculation dialog. returns - “TRUE “ if successful
IDispatch* **DyeDatabase** ()	Gets the dispatch pointer of the dye database server Interface.
Boolean **DyeDatabaseDialog** ()	Shows modal the dye database dialog. returns - “TRUE “ if successful
Boolean **ShowSplashScreen** (Boolean wait)	Shows the splash screen window. wait (IN) - if “FALSE” the methode returns immediately, if “TRUE” the mehtode waits for the end of the splash screen returns - “TRUE “ if successful
Boolean **IsSplashScreen** ()	Gets if the splash screen window id dispayed.
IlsmCalibration* **Calibration** ()	Returns a dispatch pointer to the Calibration Object.

Example:

This example starts a scan and stores the interface to the actual Image Document for a later use. Then displays the image in full screen mode.

```
Public sub StartScan()

    Dim Doc as DsRecordingDoc        ‘ define the interface object

    Set Doc = Lsm5.StartScan         ‘ Starts the scan

    Lsm5.FullScreen                  ‘ Switch to full screen mode

    …

    Set Doc = Nothing                ‘ Frees the interface object ( optional )

End Sub
```

4.3. Lsm5VbaProject

Lsm5VbaProject	
"Lsm5VbaProject" is the interface to a loaded macro project.	
Properties	
Long **MacroNumber**	Gets the number of macros in the project.
BSTR **ProjectPath**	Get/Set the path for this macro project.
BSTR **ProjectName**	Gets/Set the name of the macro project.
Long **Version**	Gets the version of the macro project.
BSTR **Autho**r	Gets the author of the macro project.
Long **Revision**	Gets the revision number of the macro project.
BSTR **Created**	Gets the date of creation of the macro project.
BSTR **Modified**	Gets the date of the last modification.
BSTR **ModifiedAuthor**	Gets the last user, who has modified the macro project.
BSTR **MacroName** (long Index)	Get the name of a macro on a specified index.
Boolean **MacroExist** (BSTR MacroName)	Check if the macro in the project exist.
BSTR **DongleKey**	Get/Set the dongle key for the run option of this macro.
BSTR **EditDongleKey**	Get/Set the dongle key for the run and edit option of this macro.
Methods	
Boolean **MacroRecordStart** (BSTR MacroName, BSTR Description)	Start the macro recording.
Void **MacroRecordEnd** (Boolean EditNow)	End the macro recording with without editing.
Void **MacroRecordCancel** ()	Cancel the macro recording.
Void **MacroRun** (BSTR MacroName)	Runs a macro from the loaded project.
Boolean **Save** ()	Saves the macro project.
Boolean **SaveAs** ()	Opens a file save dialog to save the macro project.

Void **MacroEdit** (BSTR MacroName)	Opens the editor to edit this macro.
Void **MacroStep** (BSTR MacroName)	Opens the editor to debug this macro line by line.
Boolean **ExecuteLine** (BSTR Line)	Parse and execute this line.
Void **MacroDelete** (BSTR MacroName)	Deletes a macro in the project.

4.4. Lsm5VbaDoc

Lsm5VbaDoc	
"Lsm5VbaDoc" is the macro internal representation of a macro project.	
Properties	
long **Version**	Gets the version of the macro project.
BSTR **Author**	Gets the author of the macro project.
Long **Revision**	Gets the revision nuber of the macro project.
BSTR **Created**	Gets the date of creation of the macro project.
BSTR **Modified**	Gets the date of the last modification .
BSTR **ModifiedAuthor**	Gets the last user, who has modified the macro project.
BSTR **Name**	Get/Set the name of this project
IlsmVbaApp* **Application**	Returns a dispatch pointer to the Application Object.
IlsmVbaApp* **Parent**	Returns a dispatch pointer to the Application Object.
BSTR **DongleKey**	Get/Set the dongle key for the run option of this macro.
BSTR **EditDongleKey**	Get/Set the dongle key for the run and edit option of this macro.

4.5. Lsm5Tools

Lsm5Tools	
"Lsm5Tools" is the tools interface for this server. Implements some useful functions.	
Properties	
Short **AutoChannelColor**	Not supported in this version.
Short **FastTrackSwitch**	Use the same settings (EF,PTM,...) for all tracks.
BSTR **RegStringValue** (BSTR SubKey, BSTR ValueName)	Gets a string value under a subkey.
Void **RegStringValue** (BSTR SubKey, BSTR ValueName, BSTR lpszNewValue)	Sets a string value under a subkey.
Long **RegLongValue** (BSTR SubKey, BSTR ValueName)	Gets a long value under a subkey.
Void **RegLongValue** (BSTR SubKey, BSTR ValueName, Long nNewValue)	Sets a long value under a subkey.
Double **RegDoubleValue** (BSTR SubKey, BSTR ValueName)	Gets a double value under a subkey.
Void **RegDoubleValue** (BSTR SubKey, BSTR ValueName, Double newValue)	Sets a double value under a subkey.
BSTR **MacroButtonCaption** (long Index)	Get the caption for a macro button.
Void **MacroButtonCaption** (long Index, BSTR lpszNewValue)	Set the caption for a macro button.
BSTR **MacroButtonPath** (long Index)	Get the path for a macro button.
Void **MacroButtonPath** (long Index, BSTR lpszNewValue)	Set the path for a macro button.
BSTR **MacroButtonName** (long Index)	Get the macroname for a macro button.
Void **MacroButtonName** (long Index, BSTR lpszNewValue);	Set the macroname for a macro button.

Methods	
BSTR **UserPath** ()	Get the users default path.
Void **MsgBeep** (short type)	Perform a Message Beep.
BSTR **MakeFilterSetName** (long Channel)	Creates a filtername for this channel.
BSTR **MakeMonitorFilterSetName** (long Channel)	Creates a monitor filter name for this channel (0,1)
BSTR **MakePmtName** (long Channel)	Creates a pointdetector name for this channel.
BSTR **MakePinholeName** (long Channel)	Creates a pinhole name for this channel.
BSTR **MakeAmplifierName** (long Channel)	Creates an amplifier name for this channel.
BSTR **MakeIntegratorName** (long Channel)	Creates an integrator name for this channel.
BSTR **MakeBeamSplitterName** (long Index)	Creates a beamsplitter name for this index.
BSTR **MakeChannelName** (long Channel)	Creates a channel name for this channel.
BSTR **MakeScanModeName** (long Mode)	Creates a scanmode string for this mode.
BSTR **MakeSpecialScanModeName** (long Mode)	Creates a specialscanmode string for this mode.

Boolean **ResetRecordData** ()	Reset all recording data to actual hardware.
Boolean **CheckRecordData** ()	Checks the recording against the hardware.
BSTR **MakeIlluminationChannelName** (long Channel)	Returns the name for this channel.
Boolean **RegCreateKey** (BSTR SubKey)	Create the subkey.
Boolean **RegExistKey** (BSTR Subkey)	Check if this subkey exists.
Long **RegCountSubKeys** (BSTR SubKey)	Returns the number of keys under a subkey.
BSTR **RegSubkeyName** (long Index, BSTR SubKey)	Gets the name of a key under a subkey by index.
Boolean **RegDeleteKey** (BSTR SubKey)	Deletes a subkey.
Boolean **SaveConfigurationSetting** (DsTrack* track, BSTR ConfigurationName)	Save the settings for a track under a subkey.
Boolean **LoadConfigurationSetting** (DsTrack* Track, BSTR ConfigurationName)	Load the settings for a track from a subkey.
Void **ConvertSeconds** (double SecondsIn, short* Hours, short* Minutes, short* Seconds, short* MilliSeconds, short* MicroSeconds)	Convert a great number of seconds into hour, minutes.
Boolean **SaveMarkerSetting** (DsRecording* Recording, BSTR SettingName)	Save the marker settings for this recording.
Boolean **LoadMarkerSetting** (DsRecording* Recording, BSTR SettingName)	Load the marker settings for this recording.
Boolean **LoadTimerSetting** (DsRecording* Recording, BSTR SettingName)	Load the timer settings for this recording.
Boolean **SaveTimerSetting** (Dsrecording* Recording, BSTR SettingName)	Save the timer settings for this recording.
BSTR **GetSettingKey** ()	Get the registry key for settings.

BSTR **GetMacroButtonKey** ()	Get the registry key for macro buttons.
BSTR **GetDisplayKey** ()	Get the registry key for display settings.
BSTR **GetWindowPositionKey** ()	Get the registry key for window positions settings.
Boolean **MacroButtonDelete** (long Index)	Deletes a macro button.
BSTR **GetTimerKey** ()	Get the registry key for timer settings.
BSTR **GetMarkerKey** ()	Get the registry key for marker settings.
OLE_COLOR **ColorFromWavelength** (long Wavelength)	Get the RGB color for a given wavelength.
Boolean **WaitForScanEnd** (Boolean WithStop, long lTimeInSec)	Waits for scan end. Returns FALSE if timeout.
Void **CalculateOptimalPinholes** ()	Calculate and Sets the optimal pinhole diameter.
BSTR **GetRecordConfigurationKey** ()	Get the key under witch the configurations stored.
Boolean **SaveRecording** (BSTR SaveName)	Save the actual recording settings. Returns: TRUE if the operation was successful.
Boolean **LoadRecording** (BSTR SaveName)	Loads a recording as actual configuration. SaveName (IN): name under which the configuration is stored. Returns: TRUE if the operation was successful.
Boolean **DeleteTimerSetting** (BSTR ConfigName)	Deletes a stored timer setting. Returns: TRUE if the operation was successful.
Boolean **DeleteMarkerSetting** (BSTR ConfigName)	Deletes a stored marker setting. Returns: TRUE if the operation was successful.
BSTR **GetBleachConfigurationKey** ()	Get the registry key where bleach settings stored.
Boolean **LoadBleachSetting** (BSTR SaveName)	Loads a stored bleach setting. Returns: TRUE if the operation was successful.
Boolean **SaveBleachSetting** (BSTR SaveName)	Saves a bleach setting. Returns: TRUE if the operation was successful.
Boolean **DeleteBleachSetting** (BSTR SaveName)	Deletes the specified stored bleach setting. Returns: TRUE if the operation was successful.
Boolean **LoadDefaultConfiguration** ()	Loads the default configuration for the current user. Returns: TRUE if the operation was successful.

Boolean **IsFileExist** (BSTR Filename, short Mode)	Checks, exists the given file. Returns: TRUE if the file exists.
Boolean **DisplayMessage** (enumMessageType MessageType, BSTR Message, Boolean* pVal)	Shows a message dialog Returns: Return value of the dialog.
Boolean **LoadHardwareBootMode** (long* pBootMode)	Gets the last saved boot mode of the system.
Boolean **SaveHardwareBootMode** (long BootMode)	Saves the last boot mode of the system.
long **GetSavedTimerSettingNumber** ()	Gets the number of saved timer settings.
BSTR **GetSavedTimerSettingName** (long Index)	Gets the name of the saved timer setting.
long **GetSavedMarkerSettingNumber** ()	Gets the number of saved marker settings.
BSTR **GetSavedMarkerSettingName** (long Index)	Gets the name of the saved marker setting.
long **GetSavedBleachSettingNumber** ()	Gets the number of saved bleach settings.
BSTR **GetSavedBleachSettingName** (long Index)	Gets the name of the saved bleach setting.
Boolean **ShowTubusSliderDialog** (long Position)	Shows the modal dialog for tube slider selection. Returns: TRUE if the operation was successful.

Example:

This example starts a scan and wait for the scan end.

```
Public sub StartScan()
        Dim tools as Lsm5Tools

        Set tools = Lsm5.Tools                  ‘ assigns the Lsm5Tools object.

        Lsm5.StartScan                          ‘ Starts the scan

        tools.WaitForScanEnd FALSE,10           ‘ Wait for end, without send a stop, max. 10 sec

        Set tools = Nothing

End Sub
```

4.6. Lsm5Options

Lsm5Options	
"Lsm5Options" This is the interface to the program options. For a complete description, see DS interface description	
Properties	
Boolean **SavePromptOnCloseWindow**	"SavePromptOnCloseWindow" species whether a message box is displayed when an unsaved modified image document window is about to be closed or not.
Boolean **ShowChannelsToolbar**	If "ShowChannelsToolbar" is unequal zero the "Channels" dialog bar is visible when a new image window is opened.
Boolean **ShowZoomToolbar**	If "ShowZoomToolbar" is unequal zero the "Zoom" dialog bar is visible when a new image window is opened.
Boolean **ShowSliceToolbar**	If "ShowSliceToolbar" is unequal zero the "Slice" dialog bar is visible when a new image window is opened. Note that the "Slice" dialog bar is never visible if the scan memory does not contain a stack (z-direction) and not a time series.
Boolean **ShowOverlayToolbar**	If "ShowOverlayToolbar" is unequal zero the "Overlay" dialog bar is visible when a new image window is opened.
Boolean **StatusDisplayInImageWindow**	If "StatusDisplayInImageWindow" is unequal zero the image information bar on the left border of the image window is visible when a new image window is opened.
Boolean **StatusDisplayZoom**	If "StatusDisplayZoom" is unequal zero the scanner zoom factor is displayed in the image information bar on the left border of the image window.
Boolean **StatusDisplayVoxelSize**	If "StatusDisplayVoxelSize" is unequal zero the voxel size of the scanned image in microns is displayed in the image information bar on the left border of the image window.
Boolean **StatusDisplayScanField**	If "StatusDisplayScanField" is unequal zero the total size of the scanned image in microns is displayed in the image information bar on the left border of the image window.
Boolean **StatusDisplayPinhole**	If "StatusDisplayPinhole" is unequal zero the diameter for all pinholes that where used during the scan operation is displayed in the image information bar on the left border of the image window.
Boolean **StatusDisplayObjective**	If "StatusDisplayObjective" is unequal zero the name of the objective that was used during the scan operation is displayed in the image information bar on the left border of the image window.
Boolean **StatusDisplayLasers**	If "StatusDisplayLasers" is unequal zero the AOTF-wavelengths and transmission percentage for all AOTF-lines that where used during the scan operation are listed in the image information bar on the left border of the image window.
Boolean **StatusDisplayScanSpeed**	If "StatusDisplayScanSpeed" is unequal zero the pixel integration time that was used during the scan operation is displayed in the image information bar on the left border of the image window.

Boolean **StatusDisplayDetectorGain**	If “StatusDisplayDetectorGain” is unequal zero the high voltage for all photo multipliers that where used during the scan operation are listed in the image information bar on the left border of the image window.
Boolean **StatusDisplayAmplifierGain**	If “StatusDisplayAmplifierGain” is unequal zero the gain of the detector amplifiers for all detectors that where used during the scan operation are listed in the image information bar on the left border of the image window.
Boolean **StatusDisplayAmplifierOffset**	If “StatusDisplayAmplifierOffset” is unequal zero the offset of the detector amplifiers for all detectors that where used during the scan operation are listed in the image information bar on the left border of the image window.
Boolean **StatusDisplayScanMode**	If “StatusDisplayScanMode” is unequal zero the scan mode (“Plane” , “Stack”, ...) is displayed in the image information bar on the left border of the image window.
Boolean **StatusDisplayBeamSplitters**	If “StatusDisplayBeamSplitters” is unequal zero the positions of all beam spliter that where used during the scan operation are listed in the image information bar on the left border of the image window.
Boolean **StatusDisplayFilters**	If “StatusDisplayFilters” is unequal zero the positions of the detector filters for all detectors that where used during the scan operation are listed in the image information bar on the left border of the image window.
Boolean **StatusDisplayProcessingSummary**	If “StatusDisplayProcessingSummary” is unequal zero the image processing operations that have been applied to the scan memory are listed in the image information bar on the left border of the image window.
Boolean **StatusDisplayPosition**	If “StatusDisplayPosition” is unequal zero the horizontal and vertical scanner offsets are displayed in the image information bar on the left border of the image window.
Boolean **PrintStatusDisplay**	If “PrintStatusDisplay” is unequal zero the image information display is enabled in the print preview when a new image window is opened.
Boolean **StatusPrintZoom**	If “StatusPrintZoom” is unequal zero the scanner zoom factor is displayed in the image information display for printing.
Boolean **StatusPrintVoxelSize**	If “StatusPrintVoxelSize” is unequal zero the voxel size of the scanned image in microns is displayed in the image information display for printing.
Boolean **StatusPrintScanField**	If “StatusPrintScanField” is unequal zero the total size of the scanned image in microns is displayed in the image information display for printing.
Boolean **StatusPrintPinhole**	If “StatusPrintPinhole” is unequal zero the diameter for all pinholes that where used during the scan operation is displayed in the image information display for printing.
Boolean **StatusPrintObjective**	If “StatusPrintObjective” is unequal zero the name of the objective that was used during the scan operation is displayed in the image information display for printing.
Boolean **StatusPrintLasers**	If “StatusPrintLasers” is unequal zero the AOTF-wavelengths and transmission percentage for all AOTF-lines that where used during the scan operation are listed in the image information display for printing.

Boolean **StatusPrintScanSpeed**	If “StatusPrintScanSpeed” is unequal zero the pixel integration time that was used during the scan operation is displayed in the image information display for printing.
Boolean **StatusPrintDetectorGain**	If “StatusPrintDetectorGain” is unequal zero the high voltage for all photo multipliers that where used during the scan operation are listed in the image information display for printing.
Boolean **StatusPrintAmplifierGain**	If “StatusPrintAmplifierGain” is unequal zero the gain of the detector amplifiers for all detectors that where used during the scan operation are listed in the image information bar in the image information display for printing.
Boolean **StatusPrintAmplifierOffset**	If “StatusPrintAmplifierOffset” is unequal zero the offset of the detector amplifiers for all detectors that where used during the scan operation are listed in the image information bar in the image information display for printing.
Boolean **StatusPrintScanMode**	If “StatusPrintScanMode” is unequal zero the scan mode (“Plane” , “Stack”, ...) is displayed in the image information bar on the left border of the image window.
Boolean **StatusPrintBeamSplitters**	If “StatusPrintBeamSplitters” is unequal zero the positions of all beam splitters that where used during the scan operation are listed in the image information display for printing.
Boolean **StatusPrintFilters**	If “StatusPrintFilters” is unequal zero the positions of the detector filters for all detectors that where used during the scan operation are listed in the image information display for printing.
Boolean **StatusPrintProcessingSummary**	If “StatusPrintProcessingSummary” is unequal zero the image processing operations that have been applied to the scan memory are listed in the image information display for printing.
Boolean **StatusPrintPosition**	If “StatusPrintPosition” is unequal zero the horizontal and vertical scanner offsets are displayed in the image information display for printing.
Boolean **StatusBarImageSize**	If “StatusBarImageSize**”** is unequal zero the image size in pixels is displayed in the image status bar at the lower border of the image window.
Boolean **StatusBarNumberChannels**	If “StatusBarNumberChannels**”** is unequal zero the number of data channel is displayed in the image status bar at the lower border of the image window.
Boolean **StatusBarReconstructed**	If “StatusBarReconstructed**”** is unequal zero the string “Raw Image Data” or “Reconstruction” is displayed in the image status bar at the lower border of the image window depending on whether the image is a reconstruction or not.
Boolean **StatusBarBitsPerPixel**	If “StatusBarBitsPerPixel**”** is unequal zero the number of bits per pixel is displayed in the image status bar at the lower border of the image window.
Boolean **StatusBarPixelIntensity**	If “StatusBarPixelIntensity**”** is unequal zero the intensity of the pixel under the mouse cursor is displayed for all channels in the image status bar at the lower border of the image window. The mouse cursor position in image coordinates is displayed too. If the mouse cursor is not located over the image no intensity string is displayed.

Boolean **StatusBarZoom**	If "StatusBarZoom**"** is unequal zero the display zoom factor (not the scanner zoom) is displayed in the image status bar at the lower border of the image window.
Boolean **StatusBarPalette**	If "StatusBarPalette**"** is unequal zero the name of the currently used image palette is displayed in the image status bar at the lower border of the image window.
Boolean **DatabaseListAutoArrange**	If "DatabaseListAutoArrange**"** is unequal zero the width of the columns in the database list display windows are adjusted automatically. In the other case the user can adjust the columns width with the mouse.
Boolean **DatabaseListName**	If "DatabaseListName**"** is unequal zero the column "Name" is visible in the database list display windows. This column displays the string specified in the property "Name" of the "DsRecording" object.
Boolean **DatabaseListDescription**	If "DatabaseListDescription" is unequal zero the column "Description" is visible in the database list display windows. This column displays the string specified in the property "Description" of the "DsRecording" object.
Boolean **DatabaseListNotes**	If "DatabaseListNotes" is unequal zero the column "Notes" is visible in the database list display windows. This column displays the string specified in the property "Notes" of the "DsRecording" object.
Boolean **DatabaseListDate**	If "DatabaseListDate**"** is unequal zero the column "Date/Time" is visible in the database list display windows. This column displays the time when the scan operation was started. This value can also be obtained using the "GetSample0Time" method of the "DsRecording" object.
Boolean **DatabaseListType**	If "DatabaseListType**"** is unequal zero the column "Scan Mode" is visible in the database list display windows.
Boolean **DatabaseListFormat**	If "DatabaseListFormat**"** is unequal zero the column "Pixel Depth" is visible in the database list display windows. This column displays the bits per pixel for the respective channels.
Boolean **DatabaseListStackSize**	If "DatabaseListStackSize**"** is unequal zero the column "Stack Size" is visible in the database list display windows. This column displays size of the scan field in pixels. The dimensions used during the scan operation are displayed. After image processing operations the scan memory dimensions can be changed.
Boolean **DatabaseListStackSizeMycrons**	If "DatabaseListStackSizeMycrons**"** is unequal zero the column "Stack Size" is visible in the database list display windows. This column displays size of the scan field in microns.
Boolean **DatabaseListPosition**	If "DatabaseListPosition**"** is unequal zero the column "Position" is visible in the database list display windows. This column displays the scanner offsets and the focus position.
Boolean **DatabaseListPixelDistance**	If "DatabaseListPixelDistance**"** is unequal zero the column "Scaling" is visible in the database list display windows. This column displays the voxel size in microns.
Boolean **DatabaseListObjective**	If "DatabaseListObjective**"** is unequal zero the column "Objective" is visible in the database list display windows.

Boolean **DatabaseListBeamSplitters**	If "DatabaseListBeamSplitters**"** is unequal zero the column "Beam Splitters" is visible in the database list display windows. This column lists the positions for all used beam splitters.
Boolean **DatabaseListLaserLines**	If "DatabaseListLaserLines**"** is unequal zero the column "Wavelength" is visible in the database list display windows. This column lists the AOTF-wavelengths and transmission percentage for all AOTF-lines that where used during the scan operation.
Boolean **DatabaseListFilters**	If "DatabaseListFilters**"** is unequal zero the column "Filters" is visible in the database list display windows. This column lists the detector filter positions for all detectors that where used during the scan operation.
Boolean **DatabaseListPinhole**	If "DatabaseListPinhole**"** is unequal zero the column "Pinhole" is visible in the database list display windows. This column lists the diameter for all pinholes that where used during the scan operation.
Boolean **DatabaseListProcessingSummary**	If "DatabaseListProcessingSummary**"** is unequal zero the column "Processing Summary" is visible in the database list display windows. This column lists the image processing operations that have been applied to the scan memory.
Boolean **DatabaseListProcessedSize**	If "DatabaseListProcessedSize**"** is unequal zero the column "Processed Size" is visible in the database list display windows. This column displays the scan memory size in pixels after all image processing operations. This is the size of the image in the image file.
Boolean **DatabaseListImagePathname**	If "DatabaseListImagePathname**"** is unequal zero the column "Path Name" is visible in the database list display windows. This column displays the path name of the image file relative to the directory of the database file.
Boolean **DatabaseGalleryName**	If "DatabaseGalleryName**"** is unequal zero the name of the image is displayed underneath the thumbnail in the database gallery display windows. This column displays the string specified in the property "Name" of the "DsRecording" object.
Boolean **DatabaseGalleryDate**	If "DatabaseGalleryDate**"** is unequal zero the time when the scan operation was started is displayed underneath the thumbnail in the database gallery display windows. This value can also be obtained using the "GetSample0Time" method of the "DsRecording" object.
Boolean **DatabaseGalleryType**	If "DatabaseGalleryType**"** is unequal zero the scan type used to record the scan data is displayed underneath the thumbnail in the database gallery display windows.
Boolean **DatabaseGalleryFormat**	If "DatabaseGalleryFormat**"** is unequal zero the bits per pixel for the respective channels are displayed underneath the thumbnail in the database gallery display windows.
Boolean **DatabaseGalleryStackSize**	If "DatabaseGalleryStackSize**"** is unequal zero the size of the scan field in pixels is displayed underneath the thumbnail in the database gallery display windows. The dimensions used during the scan operation are displayed. After image processing operations the scan memory dimensions can be changed.

Boolean **DatabaseGalleryProcessedSize**	If “DatabaseGalleryProcessedSize**”** is unequal zero the size of the scan memory in pixels after all image processing operations is displayed underneath the thumbnail in the database gallery display windows.

Boolean **DatabaseSeparatPath**	If “DatabaseSeparatPath” is unequal zero a separate path is used and remembered for the “Open Database” and “New Database” dialogs instead of the application path.
Boolean **DatabaseReusePath**	If “DatabaseSeparatPath” is unequal zero and “DatabaseReusePath**”** is unequal zero the most recently used database path is remembered at program shutdown and reused after restart of the program.
Boolean **ImportSeparatPath**	If “ImportSeparatPath” is unequal zero a separate path is used and remembered for the “Import” dialog instead of the application path.
Boolean **ImportReusePath**	If “ImportSeparatPath” is unequal zero and “ImportReusePath**”** is unequal zero the most recently used import path is remembered at program shutdown and reused after restart of the program.
Boolean **ExportSeparatPath**	If “ExportSeparatPath” is unequal zero a separate path is used and remembered for the “Export” dialog instead of the aplication path.
Boolean **ExportReusePath**	If “ExportSeparatPath” is unequal zero and “ExportReusePath**”** is unequal zero the most recently used export path is remembered at program shutdown and reused after restart of the program.
BSTR **DatabaseStartPath**	If “DatabaseSeparatPath” is unequal zero and “DatabaseReusePath**”** is equal zero the string in “DatabaseStartPath” is used as the start path for the “Open Database” and “New Database” dialogs after program start.
BSTR **ImportStartPath**	If “ImportSeparatPath” is unequal zero and “ImportReusePath**”** is equal zero the string in “ImportStartPath” is used as the start path for the “Import” dialog after program start.
BSTR **ExportStartPath**	If “ExportSeparatPath” is unequal zero and “ExportReusePath**”** is equal zero the string in “ExportStartPath” is used as the start path for the “Export” dialog after program start.
Long **AutosaveCount**	If “UseAutosaveName” is unequal zero an image name is generated form “AutoSaveBaseName” by appending the number “AutosaveCount” when an image is about to be saved to database. The image data are written to the database “AutoSaveDatabaseName“ and the created name is written to the name column in the “Recordings” table of the database. After the save operation was started “AutosaveCount” is incremented by one. If the database doesn’t exists the “Save As” dialog is displayed. If “UseAutosaveName” is zero the “Save As” dialog is displayed regardless of the other properties.
Boolean **UseAutosaveName**	see “AutosaveCount”
BSTR **AutoSaveBaseName**	see “AutosaveCount”
BSTR **AutoSaveDatabaseName**	see “AutosaveCount”
Boolean **WarnOverwriteImageInDatabase**	If “WarnOverwriteImageInDatabase” is unequal zero a warning is displayed when the user try to overwrite (save to database) the old data of a modified image.

Boolean **ScanInformationObjective**	If “ScanInformationObjective” is a flag used by the user interface to store the information whether the objective name should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ScanInformationPixelDepth**	If “ScanInformationPixelDepth” is a flag used by the user interface to store the information whether the number of bytes per pixel should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ScanInformationStackSize**	If “ScanInformationStackSize” is a flag used by the user interface to store the information whether the size of the scan field in microns should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ScanInformationScaling**	If “ScanInformationScaling” is a flag used by the user interface to store the information whether the voxel size in microns should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ScanInformationScanTime**	If “ScanInformationScanTime” is a flag used by the user interface to store the information whether the total scan time should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ScanInformationLaser**	If “ScanInformationLaser” is a flag used by the user interface to store the information whether the list of lasers and their state should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ScanInformationWavelength**	If “ScanInformationWavelength” is a flag used by the user interface to store the information whether the AOTF wavelengths and their percentage should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ScanInformationPosition**	If “ScanInformationPosition” is a flag used by the user interface to store the information whether the stage and focus position should be displayed in the scan information window or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **BeepBeep**	If “BeepBeep” is a flag used by the user interface to store the information when a “Beep” should be generated when the user interface has detected an error at an OLE call to other program parts. The flag is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ShutdownLaserOffOnExit**	If “ShutdownLaserOffOnExit” is a flag used by the user interface to store the information whether the lasers should be switched off an program shut down or not. The flag is written to the registry at program shutdown and reloaded at the next program start.

Boolean **StartupDontShowLogo**	If “StartupDontShowLogo” is a flag used by the user interface to store the information whether logo window should be suppressed on program start or not. The flag is written to the registry at program shutdown and reloaded at the next program start.
BSTR **StartupConfiguration**	“StartupConfiguration” is a character string where the user interface can store a name for a device configuration which should be used at the start of the program. The string is written to the registry at program shutdown and reloaded at the next program start.
Boolean **ReuseEnabled**	
Boolean **CropEnabled**	
Boolean **ConfigLoadObjective**	
Boolean **ConfigLoadFrameSize**	
Boolean **ConfigLoadSpeed**	
Boolean **ConfigLoadDataDepth**	
Boolean **ConfigLoadScanDirection**	
Boolean **ConfigLoadAverage**	
Boolean **ConfigLoadPinholeDiameter**	
Boolean **ConfigLoadDetectorGainAndOffset**	
Boolean **ConfigLoadZoom**	
boolean **ConfigLoadRotationAndOffset**	
Boolean **TimeIntervallnsteadOfCycleDelay**	

Methods	
void **Apply** ()	Changes to the properties of the “Options” object that affects the appearance of already open windows do not force a redraw and reshape of these windows. After the properties where changed one should call “Apply” to force the redraw and reshape operation.
Boolean **IsSaveModelSubdirectory** ()	There are different methods to store image files. “IsSaveModelSubdirectory” returns a value unequal zero if the actual selected method says that the image files should be stored in the sub-directory “Images” of the database path.
void **SetSaveModelSubdirectory** ()	There are different methods to store image files. “SetSaveModelSubdirectory” forces that the method to store the image files in the sub-directory “Images” of the database path should be used.
Boolean **IsSaveModelSameDirectory** ()	There are different methods to store image files. “IsSaveModelSameDirectory” returns a value unequal zero if the actual selected method says that the image files should be stored in the same directory as the database.
Void **SetSaveModelSameDirectory** ()	There are different methods to store image files. “SetSaveModelSameDirectory” forces that the method to store the image files in the same directory as the database should be used.
Boolean **IsSaveModelCreateSubdirectory** ()	There are different methods to store image files. “IsSaveModelCreateSubdirectory” returns a value unequal zero if the actual selected method says that the image files should be stored in the sub-directory of the database path. The sub-directory has the same name as the database (without drive name and directory name).
Void **SetSaveModelCreateSubdirectory** ()	There are different methods to store image files. “SetSaveModelCreateSubdirectory” forces that the method to store the image files in the subdirectory of the database should be used. The sub-directory has the same name as the database (without drive name and directory name).
Void **SetDatabaseStartViewForm** ()	“SetDatabaseStartViewForm” causes that new opened databases are initially displayed in the “Form” view.
Boolean **IsDatabaseStartViewList** ()	“IsDatabaseStartViewList” returns a value unequal zero if a new opened database is displayed initially in the “List” view.
Void **SetDatabaseStartViewList** ()	“SetDatabaseStartViewList” causes that new opened databases are initially displayed in the “List” view.
Boolean **IsDatabaseStartViewGallery** ()	“IsDatabaseStartViewGallery” returns a value unequal zero if a new opened database is displayed initially in the “Gallery” view.
Void **SetDatabaseStartViewGallery** ()	“SetDatabaseStartViewGallery” causes that new opened databases are initially displayed in the “Gallery” view.
Boolean **IsInitalRecordsetFirst** ()	“IsInitalRecordsetFirst” returns a value unequal zero if a new opened database is ever displayed with the first recordset is visible.

Void **SetInitalRecordsetFirst** ()	“SetInitalRecordsetFirst” causes that new opened databases are initially displayed so that the first recordset is visible.
Boolean **IsInitalRecordsetMiddle** ()	“IsInitalRecordsetMiddle” returns a value unequal zero if a new opened database is ever displayed with the middle recordset is visible.
Void **SetInitalRecordsetMiddle** ()	“SetInitalRecordsetMiddle” causes that new opened databases are initially displayed so that the middle recordset is visible.
Boolean **IsInitalRecordsetsetLast** ()	“IsInitalRecordsetsetLast” returns a value unequal zero if a new opened database is ever displayed with the last recordset is visible.
Void **SetInitalRecordsetLast** ()	“SetInitalRecordsetLast” causes that new opened databases are initially displayed so that the last recordset is visible.
Boolean **IsLicense** (BSTR Option)	“IsLicense” checks the given license key. Returns TRUE if the key is valid.
Boolean **IsLicenseValid** (eLicense Option)	“IsLicenseValid” checks the license for this option. Returns TRUE if the license for this option is valid.
Boolean **IsLicenseOptionValid** (eLicensePackage Option)	“IsLicenseValid” checks the license for this package. Returns TRUE if the license for this package is valid.

Example:

This example disables the “Apply” Button in the image display.

```
Sub DisableReuse()
    Dim options As Lsm5Options

    Set options = Lsm5.options          ‘ assign the Lsm5Options object

    options.ReuseEnabled = False        ‘ sets the apply value to false.

    options.Apply                       ‘ activate this new options.

    Set options = Nothing

End Sub
```

4.7. Lsm5Info

Lsm5Info	
"Lsm5Info" This is the interface to get some informations about the program and devices.	
Methods	
Boolean **NumberOfPmtsInSystem** (long* NormalPmt, long* MonitorPmt, long* TransmissionPmt)	Calculate the number of pmt types for this system.
Boolean **NumberOfExternalPmtsInSystem** (long* ExternalPmt)	Calculate the number of external pmt's for this system.
Boolean **NumberOfNDDPmtsInSystem** (long* NDDPmt)	Calculate the number of NDD pmt's for this system.
Boolean **NumberOfSPIPmtsInSystem** (long* SPIPmt)	Calculate the number of spectral imager pmt's for this system.
BSTR **VersionCp** ()	"CpVersion" is read only. The read operation returns a version string for the connected "ControlProgram" of the format "Versionhigh.VersionLow.Revision". This version string is only available after a successful call to "BootHardware". If the "DataServer" is not connected to the "ControlProgram" (that is before "BootHardware" or after "ShutDownHardware" or "Logout") an empty string is returned.
BSTR **VersionDs** ()	"VersionDs" is read only. The read operation returns a version string for the "DataServer" of the format "Versionhigh.VersionLow.Revision".
BSTR **VersionIs** ()	"VersionIs" is read only. The read operation returns a version string for the "InterfaceServer" (this Document)of the format "Versionhigh.VersionLow.Revision".
BSTR **StandType** ()	Get the stand type('Vert', 'Plan').
BSTR **StandSubType** ()	Get the stand subtype string.
BSTR **SystemDbVersion** ()	Get the control program database version string.
BSTR **FirmwareVersion** (BSTR UnitName)	Get the version string of a unit. UnitName (IN) : name of a Unit for which we want the version information.
Long **GetTubusSliderState** ()	Get the state of the Slider (TV,VIS,LSM).
Long **DongleNumber** ()	Get the existing dongle number.
BSTR **SystemVersion** ()	Get the system version string.

BSTR **Configuration** ()	Get the configuration name string.
BSTR **FCSUnit** ()	Get the FCS unit name.
Boolean I**sFCS** ()	Determine FCS mode exist
Boolean **IsLSM** ()	Determine if the LSM mode is possible.
Boolean **IsFS** ()	Is the system a FS system.
Boolean **IsPascal** ()	Is the system a Pascal system.
Boolean **IsAxioplan** ()	Is the system a Axioplan system.
Boolean **IsAxioskop** ()	Is the system a Axioskop system.
Boolean **IsAxioVert** ()	Is the system a AxioVert system.
Boolean **IsAxioplan2** ()	Is the system a Axioplan 2 system.
Boolean **IsAxioplan2i** ()	Is the system a Axioplan 2i system.
Boolean **IsAxiovert100M** ()	Is the system a Axiovert 100M system.
Boolean **IsAxiovert200M** ()	Is the system a Axiovert 200M system.
Boolean **IsPascalLaserModule** ()	Is the laser module of the system is a Pascal laser module.
Boolean **IsFiberOut** ()	Has the system a fiber out.
Boolean **IsSPI** ()	Has the system a SPI module.
Boolean **IsAnyHardwareBusy** ()	Gets if any hardware component is busy.
Short **GetSystemState** (long* State, long* Kind)	Gets the system state and the kind of the system state control.
Short **SetSystemState** (long State)	Sets the system state. (TV,VIS,LSM,FCS).

Example:

This example displays the connected Dongle number.

```
Sub ShowDongleNumber()

    Dim info As Lsm5Info

    Dim number as Long

    Set info = Lsm5.Info                ' assign the Lsm5Info object

    number = info.DongleNumber          'get the dongle number.

    MsgBox "The dongle number is: " & number

    Set info = Nothing

End Sub
```

4.8. Lsm5Constants

Lsm5Constants	
"Lsm5Constants" This is the interface of program constants used in the program. Useful to simplify the programming.	
Properties	
BSTR **StandPlan**	Get string for a plan microscope
BSTR **StandVert**	Get string for a vert microscope
BSTR **SubStandBaseport**	Get string for a baseport microscope
BSTR **SubStandSideport**	Get string for a sideport microscope
Long **MaxActiveChannelsPerTrack**	Get the max number of active channels in a track
Long **MaxActiveChannels**	Get the total max number of active channels in all tracks.
Long **MaxTracks**	Get the max number of tracks in a recording.
Long **MaxBleachTracks**	Get the max number of bleach tracks in a recording.
Methods	
BSTR **GetNothingString**()	Get the string for nothing is selected.

Example:

This example displays the maximum number of possible tracks.

```
Sub ShowMaxNumberOfTracks()

    Dim info As Lsm5Constants

    Dim number as Long

    Set info = Lsm5.Constants          ' assign the Lsm5Constants object

    number = info.MaxTracks            'get the max track number.

    MsgBox "The max. number of tracks is: " & number

    Set info = Nothing

End Sub
```

4.9. Lsm5Hardware

Lsm5Hardware	
"Lsm5Hardware" This interface enables the access to all hardware interfaces.	
Methods	
CpFilterSets* **CpFilterSets** ()	Returns the dispatch pointer to the CpFilterSet object.
CpObjectiveRevolver* **CpObjectiveRevolver** ()	Returns the dispatch pointer to the CpObjectiveRevolver object.
CpAmplifiers* **CpAmplifiers** ()	Returns the dispatch pointer to the CpAmplifiers object.
CpPinholes* **CpPinholes** ()	Returns the dispatch pointer to the CpPinholes object.
CpScancontrol* **CpScancontrol** ()	Returns the dispatch pointer to the CpScancontrol object.
CpFocus* **CpFocus** ()	Returns the dispatch pointer to the CpFocus object.
CpCollimators* **CpCollimators** ()	Returns the dispatch pointer to the CpCollimators object.
CpLamps* **CpLamps** ()	Returns the dispatch pointer to the CpLamps object.
CpLasers* **CpLasers** ()	Returns the dispatch pointer to the CpLasers object.
CpPmts* **CpPmts** ()	Returns the dispatch pointer to the CpPmts object.
CpServos* **CpServos** ()	Returns the dispatch pointer to the CpServos object.
CpShutters* **CpShutters** ()	Returns the dispatch pointer to the CpShutters object.
CpStages* **CpStages** ()	Returns the dispatch pointer to the CpStages object.
CpIntegrators* **CpIntegrators** ()	Returns the dispatch pointer to the CpIntegrators object.
CpLaserLines* **CpLaserLines** ()	Returns the dispatch pointer to the CpLaserlines object.
CpTriggers* **CpTriggers** ()	Returns the dispatch pointer to the CpTriggers object.
CpHrz* **CpHrz** ()	Returns the dispatch pointer to the CpHRZ object.

CpAomDrv* **AomDrv** ()	Returns the dispatch pointer to the CpAomDrv object.

Example:

This example selects the CpServo object for later use.

```
Sub SelectServos()

    Dim hardware    As Lsm5Hardware

    Dim servos      As CpServos

    Set hardware = Lsm5.hardware

    Set servos = hardware.CpServos

    ……..
    ……..

    Set servos = Nothing

    Set hardware = Nothing


End Sub
```

4.9.1. CpAmplifiers

CpAmplifiers	
"CpAmplifiers" This is the interface to a amplifier.	
Properties	
Double **Gain**	Get/Set the Gain of the selected amplifier.
Double **Offset**	Get/Set the Offset of the selected amplifier.
Methods	
Long **Count**()	Get the number Amplifiers in the system.
Boolean **Exist** (BSTR amplifier)	Check if this Amplifier exist.
Boolean **Select** (VARIANT amplifier)	Select a amplifier (Nr or Name).
BSTR **Name**()	Get the name of the selected amplifier.
BSTR **Status**()	Get the status of the selected amplifier.
BSTR **Summary**()	Get the summary of the selected amplifier.
Boolean **IsSelected**()	Was an amplifier successful selected.

Example:

This example displays the description(field summary) for all amplifiers connected to the system.

```
Sub DisplayAmplifierDescriptions()

    Dim amplifier As CpAmplifiers

    Dim count As Long, i As Long

    Set amplifier = Lsm5.hardware.CpAmplifiers

    count = amplifier.count

    For i = 0 To count – 1

        If (amplifier.Select(i)) Then

            MsgBox "Summary of: " & amplifier.Name & " is: " &cstr(amplifier.Summary)

        End If

    Next I

    Set amplifier = Nothing

End Sub
```

4.9.2. CpCollimators

CpCollimators	
"CpCollimators" Interface to the collimator settings.	
Properties	
Long **Value**	Get/Set the value of the selected Collimator.
Double **Position**	Get/Set the position of the selected Collimator.
Methods	
Boolean **Exist** (BSTR collimator)	Check if this Collimator exist.
Long **Count**()	Get the number of Collimators.
Boolean **Select** (VARIANT collimator)	Select a Collimator(number or name).
BSTR **Name**()	Get the name of the selected Collimator.
long **MinValue**()	Get the minimum value of the selected Collimator.
long **MaxValue**()	Get the maximum value of the selected Collimator.
BSTR **Status**()	Get the status of the selected Collimator.
BSTR **Summary**()	Get the summary of the selected Collimator.
BSTR **CollimatorName** (long Index)	Get the name of a Collimator.
Boolean **IsSelected**()	Is a collimator successful selected.
Boolean **StorePositionAsDefault()**	Store the actual position as default position.
Boolean **MoveToDefaultPosition()**	Moves to the default position.
Boolean **IsBusy ()**	Gets if the collimator is busy.
double **MinPosition**()	Get the minimum position of the selected Collimator.
double **MaxPosition**()	Get the maximum position of the selected Collimator.
long **MinWavelength**()	Get the minimum wavelength of the selected Collimator.
long **MaxWavelength** ()	Get the maximum wavelength of the selected Collimator.
Boolean **CollPositionFromWavelength** (long WaveLength, double* pPosition)	Calculates the optimal collimator position for a given wavelength.
Boolean **PHZPositionFromWavelength** (long WaveLength, double* pPosition)	Calculates the optimal pinhole z-position for a given wavelength.
Boolean **AreObjectiveDataAvailable**()	Gets if pinhole data for optimal collimator and pinhole z-position calculation are available.

Example:

This example displays the positions of all collimators connected to the system.

```
Sub DisplayCollimatorPositions()
   Dim collimator As CpCollimators
   Dim count As Long, i As Long
   Dim Result As String
   Set collimator = Lsm5.hardware.CpCollimators
   count = collimator.count
   Result = “”
   For i = 0 To count – 1
      If (collimator.Select(i)) Then
         Result = Result + "Position of Collimator : "
         Result = Result + collimator.Name
         Result = Result + " is: "
         Result = Result + CStr(collimator.Position)
         Result = Result + vbCrLf
      End If
   Next I
   MsgBox Result
  Set collimator = Nothing
End Sub
```

4.9.3. CpFilterSets

CpFilterSets	
"CpFilterSets" Interface to manipulate or set filter set methods or properties.	
Properties	
Long **FilterSetPosition**	Get/Set the actual FilterSet position by number.
BSTR **FilterSetPositionName**	Get/Set the actual FilterSet position by name.
Long **LastErrorCode**	Get the last error of this FilterSet.
Long **FilterSetKind**	Get the kind of this FilterSet.
Boolean **FilterSetIsManualChangeable**	Get TRUE if the FilterSet is manual changeable.
Methods	
Long **Count**()	Get the number of FilterSets.
Boolean **Exist** (BSTR FilterSet)	Check if this FilterSet exist.
Boolean **Select** (VARIANT FilterSet)	Select this FilterSet as actual. By number or name.
Long **FilterSetFilterCount**()	Get the number of Filters in the actual selected FilterSet.
BSTR **FilterName** (long Index)	Get the name of a Filter in the actual FilterSet.
Long **FilterPosition** (long Index)	Get the position of a Filter in the actual FilterSet.
Double **FilterTransmission** (long Index)	Get the transmission of a Filter in the actual FilterSet.
Long **FilterWaveLength** (long Index)	Get the wavelength of a Filter in the actual FilterSet.
BSTR **FilterSummary** (long Index)	Get the summary of a Filter in the actual FilterSet.
BSTR **FilterSetName**()	Get the name of the actual FilterSet.
BSTR **FilterSetStatus**()	Get the status of the actual FilterSet.
BSTR **FilterSetSummary**()	Get the summary of the actual FilterSet.
Boolean **FilterExist** (BSTR Filter)	Check if this filter in the actual FilterSet exist.
Boolean **RebootAll**()	Reboot all filtersets.
Boolean **IsSelectedFilterSet**()	Is a filterset successful selected.

Long **FilterSetIndex**()	Returns the index of the actual filter in the selected filterset.
Long **FilterWaveLengthFromName** (BSTR FilterName)	Get the wavelength of a Filter in the actual FilterSet.
Double **FilterTransmissionFromName** (BSTR FilterName)	Get the transmission of a Filter in the actual FilterSet.

Example:

This example displays all Filters in the FilterSet “HT”.

```
Sub DisplayHTFilters()
   Dim filterset As CpFilterSets
   Dim count As Long
   Dim i As Long
   Dim Result As String
   Set filterset = Lsm5.hardware.CpFilterSets
   If (filterset.Select("HT")) Then
      count = filterset.FilterSetFilterCount  ' number of filters in the set
      For i = 0 To count – 1
         Result = Result + filterset.FilterName(i)
         Result = Result + vbCrLf
      Next I
      MsgBox "The Filters in Filterset 'HT' are:" & vbCrLf & Result
   End If
   Set filterset = Nothing
End Sub
```

4.9.4. CpFocus

CpFocus	
"CpFocus" Interface to the methods and properties of a focus.	
Properties	
Double **Position**	Get/Set the focus position [µm].
Double **Stepsize**	Get/Set the manual focus step size [µm].
Double **RefractionCorrection**	Get/Set the refraction correction.
Boolean **ConnectHrzToFocus**	Get/Set HRZ – focus connection
Methods	
Boolean **SetZero**()	Zero's the focus position.
Boolean **MoveToWorkPosition**()	Move to the focus work position.
Boolean **MoveToLoadPosition**()	Move to the focus load position.
BSTR **Status**()	Get the status of the focus.
BSTR **Summary**()	Get the summary of the focus.
Boolean **Exist** (BSTR focus)	Check if this Focus exist.
Long **Count**()	Get the number of Focus objects.
Boolean **Select** (VARIANT Focus)	Select this Focus as actual. By number or name.
Boolean **IsSelected**()	Was a focus successful selected.
Void **MoveRelative** (double Offset)	Move the focus relative to this current position.
Double **MinStackStep()**	Gets the minimal step size.
Boolean **IsBusy()**	Gets if the focus is busy.

Example:

This example moves the focus relative 120 µm.

```
Sub MoveFocusRelative()
   Dim focus As CpFocus
   Set focus = Lsm5.hardware.CpFocus
   If (focus.Select(0)) Then                    ‘ select the focus if more then 1 exist
      focus.MoveRelative = 120
   End If
   Set focus = Nothing
End Sub
```

4.9.5. CpIntegrators

CpIntegrators	
"CpIntegrators" Interface to the methods and properties of a integrator.	
Methods	
Long **Count** ()	Get the number of integrators in this system.
Boolean **Exist** (BSTR Integrator)	Check if this integrator exist.
Boolean **Select** (VARIANT Integrator)	Select a integrator by name or index.
BSTR **Name** ()	Get the name of the selected integrator.
BSTR **Summary** ()	Get the summary of the selected integrator.
BSTR **Status** ()	Get the status of the selected integrator.
Boolean **IsSelected** ()	Is a integrator successful selected.

Example:

This example gets the status of all integrators.

```
Sub DisplayIntegratorStatus()
   Dim integrator As CpIntegrators
   Dim count As Long, i As Long
   Dim Result As String
   Set integrator = Lsm5.hardware.CpIntegrators
   count = integrator.count
   For i = 0 To count – 1
      If (integrator.Select(i)) Then
         Result = Result + integrator.Name
         Result = Result + vbTab
         Result = Result + integrator.Status
         Result = Result + vbCrLf
      End If
   Next I
   Set integrator = Nothing
   MsgBox "Status of Integrators: " & vbCrLf & Result
End Sub
```

4.9.6. CpLamps

CpLamps	
"CpLamps" Interface to the methods and properties of a lamp.	
Properties	
Boolean **OnOff**	Get/Set the selected Lamp on/off.
Boolean **Remote**	Get/Set remote switch of the selected Lamp.
Double **Voltage**	Get/Set the power of the selected Lamp.
Boolean **Lamp3200K**	Get/Set the 3200K switch of the selected Lamp.
Double **IntensityPercent**	Get/Set the Intensity of the selected Lamp in percent.
Methods	
Boolean **Exist** (BSTR lamp)	Check if this Lamp exist.
Long **Count**()	Get the number of Lamps.
Boolean **Select** (VARIANT lamp)	Select a Lamp (by name or number).
BSTR **Name**()	Get the name of the selected Lamp.
BSTR **Status**()	Get the status of the selected Lamp.
BSTR **Summary**()	Get the summary of the selected Lamp.
Double **MinInputVoltage**()	Get the minimal input voltage of the selected Lamp.
Double **MaxInputVoltage**()	Get the maximal input voltage of the selected Lamp.
Double **MinOutputVoltage**()	Get the minimal output voltage of the selected Lamp.
Double **MaxOutputVoltage**()	Get the maximal output voltage of the selected Lamp.
Boolean **IsSelected**()	Was a lamp successful selected.

Example:

This example switches all lamps off.

```
Sub SwitchLampsOff()
   Dim lamps As CpLamps
   Dim count As Long
   Dim i As Long
   Set lamps = Lsm5.hardware.CpLamps
   count = lamps.count
   For i = 0 To count – 1
      If (lamps.Select(i)) Then
         lamps.OnOff = False
      End If
   Next i
   Set lamps = Nothing
End Sub
```

4.9.7. CpLaserLines

CpLaserLines	
"CpLaserLines" Interface to the methods and properties of a laserline.	
Methods	
Long **Count** ()	Get the number of laserlines in this system.
Boolean **LineInfo** (long Index, long* WaveLength, double* Attenuation, short* OnOff, BSTR* LaserName)	Get information about the laser line of this index.
Double **Attenuation** (long WaveLength)	Get the attenuation of a wavelength.
Void **Attenuation** (long WaveLength, double newValue)	Set the attenuation of a wavelength to value.
Boolean **OnOff** (long WaveLength)	Get the state of a wavelength.
Void **OnOff** (long WaveLength, boolean bNewValue)	Set the state of a wavelength.
Boolean **IsLaserOn** (long Wavelength)	Check if the laser for this line is on or off.
Long **AttenuatorType** (long Wavelength)	Get the attenuator type for this wavelength.
long **SwitchableWavelengthsPerIndex** (long index)	Get the number of switchable wavelength of this index.

Example:

This example switches all LaserLines on and set the attenuation to 50 %.

```
Sub SwitchAllLaserLinesTo50()
   Dim laserlines As CpLaserLines
   Dim wavelength As Long
   Dim attenuation As Double
   Dim onoff As Integer
   Dim laser As String
   Dim count As Long
   Dim i As Long
   Set laserlines = Lsm5.hardware.CpLaserLines
   count = laserlines.count
   For i = 0 To count – 1
      laserlines.LineInfo i, wavelength, attenuation, onoff, laser
      laserlines.attenuation(wavelength) = 50
      laserlines.onoff(wavelength) = True
   Next i
   Set laserlines = Nothing
End Sub
```

4.9.8. CpScanControl

CpScanControl	
"CpScanControl" Interface to the methods and properties of the scancontrol. Most used to get the actual state of the scan process.	
Properties	
double **PhaseShiftX** ()	Get the phase shift X correction.
void **PhaseShiftX** (double val)	Set the phase shift X correction.
double **PhaseShiftY** ()	Get the phase shift Y correction.
void **PhaseShiftY** (double val)	Set the phase shift Y correction.
Methods	
Void **StopScan** ()	Stops the current scan process.
Boolean **RebootDsp** ()	Reboot the DSP. Not supported in this version.
Boolean **IsScanning** ()	Check the a scan process is active (bleach or grap).
Double **CalculateSampleObservationTime** (long speed, long trackindex)	Calculate the time for the speed index value.
Long **ScanSpeed** (double dSampleObservationTime, long Trackindex)	Get the speed index for the given time.
Double **TotalTimePerFrame** ()	Get the total scan time.
Double **MaxScanfieldHeightZoom1** (long lVisible)	Get the height if the zoom is 1.
Double **MaxScanfieldWidthZoom1** (long lVisible)	Get the width if the is 1.
Boolean **IsBleaching** ()	Check if the bleach process is active.
Boolean **IsGrapping** ()	Check if the grap process is active.
BSTR **ArePinholesAdjusted** (short* Adjusted)	Checks if the pinholes adjusted, Returns the display string.

Boolean **AutoDdsCorrection** ()	Starts the auto dds correction process.
Void **FinishScan** ()	Finish the current scan . Ends on the end of the current image. For continues average mode.
Void **SendTriggerOut** (long Index)	Sends a trigger out.
Void **EnableTriggerInEvent** (long Index, short Enable)	Enables or disables a trigger in.
long **GetScanState**()	Gets the scan state.

Example:

This example stops a scan if a scan is running.

```
Sub StopScan()

   Dim scancontrol As CpScancontrol

   Set scancontrol = Lsm5.hardware.CpScancontrol

   If (scancontrol.IsScanning) Then

      scancontrol.StopScan

   End If

   Set scancontrol = Nothing

End Sub
```

4.9.9. CpLasers

CpLasers	
"CpLasers" Interface to the methods and properties of a laser.	
Properties	
Long **State**	Get/Set the state of the selected Laser.
Double **Power**	Get/Set the power of the selected Laser.
Double **PowerWatt**	Get/Set the power of the selected Laser in Watt.
Double **PumpPowerWatt**	Get/Set the pump power of the selected Laser in Watt.
Methods	
Long **Count**()	Get the number of Lasers in the system.
Boolean **Exist** (BSTR laser)	Check if this Laser exist.
BSTR **Name**()	Get the name of the selected Laser.
BSTR **Status**()	Get the state of the selected Laser.
BSTR **Summary**()	Get the summery of the selected Laser.
BSTR **Model**()	Get the model of the selected Laser.
Long **LineCount**()	Get the number of lines of the selected Laser.
Long **WaveLength** (long Index)	Get the wavelength of a line of the selected Laser.
Double **MaximumPower**()	Get the maximum power of the selected Laser.
Boolean **IsPowerChangeable**()	Get if the selected Lasers power is changeable.
Double **TubeCurrent**()	Get the tube current of the selected Laser.
Boolean **Select** (VARIANT Laser)	Select a Laser (by name or number).
Boolean **IsSelected**()	Is a laser successful selected.
Double **MinimumPowerPercent** ()	Get the minimal power of the selected laser [%].
Boolean **IsWavelengthChangeable** ()	Determine, is this laser a wavelength changeable Type.
double **MinimumPowerWatt** ()	Get the minimal power of the selected laser in Watt.

double **BandWidth** ()	Get the bandwidth of the selected laser.
double **MinPumpPowerWatt** ()	Get the minimal pump power of the selected laser in Watt.
double **MaxPumpPowerWatt** ()	Get the maximal pump power of the selected laser in Watt.
double **ActualPumpPowerWatt** ()	Get the pump power of the selected laser in Watt.

Example:

This example switches all Lasers off

```
Sub SwitchAllLasersOff()
   Dim laser As CpLasers
   Dim count As Long
   Dim i As Long
   Set laser = Lsm5.hardware.CpLasers
   count = laser.count
   For i = 0 To count – 1
      If (laser.Select(i)) Then
         laser.State = eLaserOff
      End If
   Next i
   Set laser = Nothing
End Sub
```

4.9.10. CpObjectiveRevolver

CpObjectiveRevolver	
"CpObjectiveRevolver" Interface to the methods and properties of a objective revolver. Normaly only one objective revolver exists in the system.	
Properties	
Long **RevolverPosition**	Get/Set the position of the actual Objective Revolver by number.
BSTR **RevolverPositionName**	Get the name of the actual Objective Revolver / Set the revolver by name
Double **RevolverPositionMagnification**	Get the magnification of the actual Objective Revolver/ Set the revolver by magnification.
Methods	
Long **Count**()	Get the number Objective Revolvers in the system.
BSTR **Name** (long Index)	Get the name of a Objective in the actual revolver.
Long **Position** (long Index)	Get the position of a Objective in the actual revolver.
Double **Aperture** (long Index)	Get the aperture of a Objective in the actual revolver.
Double **Magnification** (long Index)	Get the magnification of a Objective in the actual revolver.
Double **FreeWorkingDistance** (long Index)	Get the workingdistance of a Objective in the actual revolver.
Boolean **Select** (VARIANT revolver)	Select a Revolver as the actual revolver.
Boolean **Reboot**()	Reboot the data of the actual revolver.
BSTR **Summary** (long Index)	Get the summary of a Objective in the actual revolver.
BSTR **RevolverSummary**()	Get the summary of the actual revolver.
Boolean **Exist** (BSTR revolver)	Check if this Revolver exist.
Long **CountObjectives**()	Get the number of objectives in the selected revolver.
Boolean **ExistObjectives** (BSTR Objective)	Checks if this objective in the selected revolver exist.
Long **PositionIndex** (long Position)	Get the index of a objective on a position.
Long **RevolverKind**()	Get the kind of this revolver (manual | coded | motorized).

Boolean **IsSelected**()	Was a revolver successful selected.

Example:

This example gets the aperture of the actual objective in the revolver.

```
Sub GetCurrentAperture()
        Dim revolver As CpObjectiveRevolver

        Dim Position as Integer

        Dim ActIndex As Integer

        Set revolver = Lsm5.Hardware.CpObjectiveRevolver

        Position = revolver.RevolverPosition          ‘ gets the actual position of a
                                                      objective.

        ActIndex = revolver.PositionIndex(Position)   ‘ detrmine the list index.

        MsgBox "Aperture = " & revolver.Aperture(ActIndex)   ‘ gets and display the aperture.

        Set revolver = Nothing

End Sub
```

4.9.11. CpPinholes

CpPinholes	
"CpPinholes" Interface to the methods and properties of a pinhole.	
Properties	
Double **Diameter**	Get/Set the Diameter for the selected pinhole.
Long **Value**	Get/Set the Value for the selected pinhole.
double **PositionX**	Get/Set the Position X for the selected pinhole.
double **PositionY**	Get/Set the Position Y for the selected pinhole.
double **PositionZ**	Get/Set the Position Z for the selected pinhole.
Methods	
Long **Count** ()	Returns the total number of pinholes.
Boolean **Select** (VARIANT pinhole)	Select this pinhole as current. Indexnumber or String.
Boolean **Exist** (BSTR pinhole)	Check if this pinhole exist.
BSTR **Name**()	Get the name of the selected pinhole.
Double **AiryUnits** (double lambda, double dDiameter)	Get the AiryUnits for the selected pinhole. If lambda is zero, the program calculates the lambda value.
Double **Zresolution** (double lambda1, double lambda2, double dDiameter)	Get the Zresolution for the selected pinhole. If the lambda values are zero, the program calculates These values.
Double **MaxDiameter** ()	Get the MaxDiameter for the selected pinhole.
Double **MinDiameter** ()	Get the MinDiameter for the selected pinhole.
Long **MaxValue** ()	Get the MaxValue for the selected pinhole.
Long **MinValue** ()	Get the MinValue for the selected pinhole.
Boolean **MoveToStoredPosition** ()	Move the selected pinhole to stored x, y and z Position.
Boolean **SavePosition** ()	Store x, y, and z Position of the selected pinhole.

BSTR **Summary** ()	Get the Summary for the selected pinhole.
Boolean **IsSelected** ()	Is a pinhole successful selected.
double **MinPositionX** ()	Get the minimum x position for the selected pinhole.
double **MaxPositionX** ()	Get the maximum x position for the selected pinhole.
double **MinPositionY** ()	Get the minimum y position for the selected pinhole.
double **MaxPositionY** ()	Get the maximum y position for the selected pinhole.
double **MinPositionZ** ()	Get the minimum z position for the selected pinhole.
double **MaxPositionZ** ()	Get the maximum z position for the selected pinhole.
boolean **IsBusyDiameter** ()	Get if the diameter is busy.
boolean **IsBusyX** ()	Get if the x position is busy.
boolean **IsBusyY** ()	Get if the y position is busy.
boolean **IsBusyZ** ()	Get if the z position is busy.
boolean **ExistsZ** ()	Get if the z position is changeable.

4.9.12. CpPmts

CpPmts	
"CpPmts" Interface to the methods and properties of a photo detector.	
Properties	
Double **Voltage**	Get/Set the Voltage for the selected Pmt.
Double **Gain**	Get/Set the Gain for the selected Pmt.
Boolean **OnOff**	Switch the selected Pmt On or Off.
Methods	
Boolean **Exist** (BSTR PmtName)	Check if this Pmt exist.
Long **Count** ()	Get the number of Pmt's in the system.
Boolean **Select** (VARIANT pmt)	Select a Pmt (by Name or Nr).
BSTR **Name** ()	Get the Name of the selected Pmt.
BSTR **Status** ()	Get the Status of the selected Pmt.
BSTR **Summary** ()	Get the Summary of the selected Pmt.
Double **MaxVoltage** ()	Get the Maximum Voltage of the selected Pmt.
Double **MinVoltage** ()	Get the Minimum Voltage of the selected Pmt.
Boolean **IsSelected** ()	Is a Pmt successful selected.
eDetectorTypeCode **DetectorType** ()	Get the detector type of the selected Pmt.
long **SpectralChannels** ()	Get the number of spectral channels of the selected Pmt.
double **SpectralBandwidth** ()	Get the bandwith of one spectral channel of the selected Pmt.

4.9.13. CpServos

CpServos	
"CpServos" Interface to the methods and properties of a servo.	
Properties	
Double **Position**	Get/Set the Position for the selected Servo.
Double **Value**	Get/Set the Value for the selected Servo.
Boolean **IsManualChangeable**	Is this servo manual changeable.
Methods	
Long **Count** ()	Get the number of servos in the system.
Boolean **Exist** (BSTR servo)	Check if this Servo exist.
Boolean **Select** (VARIANT servo)	Select a Servo (by Name or Nr).
BSTR **Name** ()	Get the Name of the selected Servo.
BSTR **Status** ()	Get the Status of the selected Servo.
BSTR **Summary** ()	Get the Summary of the selected Servo.
Double **MaxValue** ()	Get the maximum Value of the selected Servo.
Double **MinValue** ()	Get the minimum Value of the selected Servo.
Double **MaxPosition** ()	Get the maximum Position of the selected Servo.
Double **MinPosition** ()	Get the minimum Position of the selected Servo.
Boolean **IsSelected** ()	Is a servo successful selected.

4.9.14. CpShutters

CpShutters	
"CpShutters" Interface to the methods and properties of a shutter.	
Properties	
Long **Position**	Get/Set the Position for the selected Shutter.
Methods	
Long **Count** ()	Get the number of Shutters in the system.
Boolean **Exist** (BSTR shutter)	Check if this Shutter exist.
Boolean **Select** (VARIANT shutter)	Select a Shutter (by Name or Nr).
BSTR **Name** ()	Get the Name of the selected Shutter.
BSTR **Status** ()	Get the Status of the selected Shutter.
BSTR **Summary** ()	Get the Summary of the selected Shutter.
Boolean **IsMotorized** ()	Get if the selected Shutter is motorized.
Boolean **IsManualChangeable** ()	Get if the selected Shutter is manual changeable.
Boolean **IsSelected** ()	Is a shutter successful selected.

4.9.15. CpStages

CpStages	
"CpStages" Interface to the methods and properties of a stage. Normally only one stage exists.	
Properties	
Double **PositionX**	Get/Set the stage position in X direction in µm.
Double **PositionY**	Get/Set the stage position in Y direction in µm.
Double **StepSize**	Get/Set the stage step size in µm.
Double **MinMarkDistance**	Get/Set the marker catch range.
Long **MotorSpeed**	Get/Set the stage motor speed., index of possible speeds(0…2)
Short **SwitchManual** (short Joystick)	Determine if the stage or the joystick is in manual mode. 0 = Stage , 1= Joystick
Void **SwitchManual** (short Joystick, short nNewValue)	Set the stage or the joystick to manual(1) or automatic(0) mode.
Methods	
Long **Count** ()	Get the number of stages.
Boolean **SetZero** ()	Set the stage position to zero.
BSTR **Status** ()	Get the stage status.
BSTR **Summary** ()	Get the stage summary.
Boolean **Exist** (BSTR Stage)	Check if this stage exist.
Boolean **Select** (VARIANT Stage)	Select a stage (by name or number).
Boolean **IsSelected** ()	Is a stage successful selected.
Long **MarkCount** ()	Get the number of stored markers.
Short **MarkClear** (long Index)	Clears the marker at the index position.
Void **MarkClearAll** ()	Clears all markers.

Long **MarkGet** (long Index, double* Xposition, double* YPosition)	Get the marker positions at the index.
Void **MarkMoveTo** (long Index)	Move to the index position.
Long **MarkAdd** (double XPosition, double YPosition)	Add a new marker.
Long **MarkAddReleativ** (double XPosition, double YPosition)	Add a new marker.
Long **MarkGetIndex** (double Xposition, double Yposition)	Get the marker index in the near of this position.
Boolean **HrzNull** ()	Sets the HRZ position to zero.
Boolean **IsBusy** ()	Determines is the stage busy.

4.9.16. CpTriggers

CpTriggers	
"CpTriggers" Interface to the methods and properties of a trigger. Triggers are used to react on an external or internal event (input trigger)or .send an event (output trigger).	
Methods	
Long **GetTriggerInCount** ()	Get the number of input trigger.
Long **GetTriggerOutCount** ()	Get the number of output trigger.
BSTR **GetTriggerInName** (long Index)	Get the input trigger name.
BSTR **GetTriggerOutName** (long Index)	Get the output trigger name.
Boolean **TriggerInExists** (BSTR TriggerIn)	Check if the input trigger name exist.
Boolean **TriggerOutExists** (BSTR TriggerOut)	Check if the output trigger name exist.

4.9.17. CpHrz

CpHrz	
"CpHrz" Interface to the methods and properties of a HRZ. Normally only one HRZ exist.	
Properties	
Double **Position**	Get/Set the focus position.
Double **Stepsize**	Get/Set the manual focus step size.
Double **Calibrate**	Get/Set the calibration.
Methods	
BSTR **Status**()	Get the status of the HRZ.
BSTR **Summary** ()	Get the summary of the HRZ.
Boolean **Exist** (BSTR name)	Check if this HRZ exist.
Long **Count** ()	Get the number of HRZ objects.
Boolean **Select** (VARIANT HRZName)	Select this HRZ as actual.By number or name.
Boolean **IsSelected** ()	Was a HRZ successful selected.
Boolean **SetPriorityPosition** (long Position)	Sets the priority position of the HRZ.
Boolean **SetCloseCondition** ()	Sets the close condition.
Double **HrzStep** (boolean Zoom)	
Double **GetMaxPosition** ()	Gets the maximum HRZ position.
Double **GetMinPosition** ()	Gets the minimum HRZ position.
Double **GetMinCalibrate** ()	Gets the minimum calibration value.
Double **GetMaxCalibrate** ()	Gets the maximum calibration value.
Boolean **GetIsManuellCalibratable** ()	Returns TRUE if the HRZ is manual calibratible.

Boolean **Leveling** ()	Perform a HRZ leveling with the focus.
Boolean **GetIsZSectEnabled** ()	Returns TRUE if the HRZ can use Zsectioning.
Double **GetMaxScanSize** ()	Returns the maximum scansize in z direction.

4.9.18. CpAomDrv

CpAomDrv	
"CpAomDrv" This is the interface to a aom driver.	
Properties	
Long **LaserWavelength**	Get/Set the new wavelength of the selected Aom driver.
double **DriverFrequency**	Get/Set the Frequency the selected Aom driver.
long **DriverPower**	Get/Set the Power the selected Aom driver.
Methods	
Long **Count**()	Get the number of Aom drivers.
Boolean **Exist** (BSTR AomDrv)	Checks if this Aom driver exists.
Boolean **Select** (VARIANT AomDrv)	Select a AomDrv as actual (Nr or Name).
BSTR **Name**()	Get the name of a selected Aom driver.
BSTR **Status**()	Get the status of a selected Aom driver.
BSTR **Summary**()	Get the summary of a selected Aom driver.
Boolean **RebootAll** ()	Reboot all drivers.
Boolean **Store** ()	Store the changed properties of a selected Aom driver.
long **MaxLaserWavelength** ()	Gets the max possible Laser wavelength.
Long **MinLaserWavelength** ()	Gets the min possible Laser wavelength.
Double **MaxDriverFrequency** ()	Gets the max possible AOM driver frequency.
Double **MinDriverFrequency** ()	Gets the min possible AOM driver frequency.
Long **MaxDriverPower** ()	Gets the max possible AOM driver power.
Long **MinDriverPower** ()	Gets the min possible AOM driver power.
double **CalculateFrqFromWavelength** (double Wv)	Calculate the frequency from the wavelength.

4.10. Lsm5 Data

4.10.1. DsRecordingDoc

DsRecordingDoc	
"DsRecordingDoc" Interface to a scan document.	
Properties	
Double **VoxelSizeX**	
double **VoxelSizeY**	
double **VoxelSizeZ**	
double **OriginX**	
double **OriginY**	
double **OriginZ**	
VARIANT **ColorPalette**	
boolean **Mono**	
Long **ChannelColor** (long Channel)	
BSTR **ChannelName** (long Channel)	
Double **TimeStamp** (long Index)	
VARIANT **ScanLine** (long StackNumber, long ChannelNumber, long FrameNumber, long LineNumber, long* SamplesPerLine, long* BytesPerPixel)	"ScanLine" can be used to get access to the pixel data of one scan line. The Scan memory is organized in 5 dimensions: ■ Channel number ■ Stack number (time-dimension) ■ Plane number (z-dimension) ■ Line number (y-dimension) ■ Sample number (x-dimension) StackNumber (IN) - stack number for the pixel data ChannelNumber (IN) - channel number for the pixel data FrameNumber (IN) - plane number in the stack for the pixel data LineNumber (IN) - line number in the plane for the pixel data SamplesPerLine (OUT) - returns the number of samples per line (y - dimension) in the returned safe array

	BytesPerPixel (IN/OUT) - returns the number of bits per pixel (y - dimension) in the returned safe array. On input one can specify if 1 or 2 bytes per pixel should be supplied. If the internal pixel size should be used one can set "*plBytesPerPixe" to "0". Returns - a safe array witch copied pixel data from one scan line
Void **ScanLine** (long StackNumber, long ChannelNumber, long FrameNumber, long LineNumber, long* SamplesPerLine, long* BytesPerPixel, VARIANT newValue)	"ScanLine" can be used to set one scan line of pixel data. The Scan memory is organized in 5 dimensions: ■ Channel number ■ Stack number (time-dimension) ■ Plane number (z-dimension) ■ Line number (y-dimension) ■ Sample number (x-dimension) StackNumber (IN) - stack number for the pixel data ChannelNumber (IN) - channel number for the pixel data FrameNumber (IN) - plane number in the stack for the pixel data LineNumber (IN) - line number in the plane for the pixel data SamplesPerLine (IN/OUT) - returns the number of samples per line (y - dimension) in the returned safe array BytesPerPixel (IN/OUT) - returns the number of bits per pixel (y - dimension) in the returned safe array. On input one can specify if 1 or 2 bytes per pixel should be supplied. If the internal pixel size should be used one can set "*BytesPerPixel" to "0". NewValue (IN) - Reference to a safe array variant with pixel data
Methods	
DsRecording* **Recording** ()	Get the DsRecording object for the selected document.
BSTR **Title** ()	Title returns the name of the recording (image) document. If the image was loaded for a database the name is the same as specified in the “Name” field of the table “Recordings”. If the image was imported from file the name is the filename.
Boolean **CloseAllWindows** ()	“CloseAllWindows” closes all image windows of the recording (image) document. For the time being we only have one image window but that fact can be obsolete in future. The recording (image) document is not destroyed. returns - unequal zero if successful

Boolean **SwitchToLineScanDiagram** ()	If the image window of the recording document does not show a “linescan” diagram this method switches to “linescan” diagram display. returns - unequal zero if successful
Boolean **GetCurrentMousePosition** (long* C, long* T, long* Z, long* Y, long* X)	Returns the information over which Channel ‘C’, TimeChannel ‘T’, Stack ‘Z’ Position ‘Y’ and Position ‘X’ the mouse is.
Boolean **IsMousePointerOverWindow** ()	Determines, is the mouse currently over a DS Image Window.
Boolean **EnableImageWindowEvent** (enumDataserverEvents EventNumber, BOOL Enable)	This function enables or disables the event generation for the image windows. EventNumber(IN): which event to be changed Enable(IN) TRUE, FALSE Returns TRUE if the function was successful.
EnumDisplayMode **GetCurrentDisplayMode** ()	Returns the current ImageWindow display mode. See enumerations for the return value.
Boolean **SwitchToDisplayMode** (enumDisplayMode Mode)	Switches the ImageWindow to the given display mode. Mode (IN): new display mode. See enumerations for the parameter values.
Boolean **CanSwitchToDisplayMode** (enumDisplayMode Mode)	Determines, can we switch to the given display mode. Mode(IN): testing display mode. See enumerations for the parameter values.
Long **GetFrameWindowHandle** ()	Returns the handle to the image frame window as long value.
Boolean **RedrawImage** ()	Starts the redraw of the image window.
Boolean **SetHotSpot** (long Channel, long Time, long Z, long Y, long X)	Displays the image, which is determined through the parameters ‘Channel’, ‘Time’, ‘Z’, Y’ and ‘X’
Boolean **GetHotSpot** (long* Channel, long* Time, long* Z, long* Y, long* X)	Get the current Image parameters.
Long **GetDimensionX** ()	Returns the dimension of the current image in the X direction in pixels
Long **GetDimensionY** ()	Returns the dimension of the current image in the Y direction in pixels

Long **GetDimensionZ** ()	Returns the dimension of the current image in the Z direction as number
Long **GetDimensionTime** ()	Returns the dimension of the current image in the Time direction as number
Long **GetDimensionChannels** ()	Returns the dimension of the current image in the Channel direction as number
Boolean **IsDataTypeAnimation** ()	"IsDataTypeAnimation" returns TRUE if the image is an animation image.
Boolean **SetDataTypeAnimation** ()	"SetDataTypeAnimation" changes the data type of the to the animation datatype.
Boolean **IsDataTypeOriginalData** ()	"IsDataTypeOriginalData" returns TRUE, if the image window is an original scan image.
Boolean **SetDataTypeOriginalData** ()	"SetDataTypeOrginalData" sets the data type to the type : Orginal data.
Boolean I**sDataTypeCalculatedData** ()	"IsDataTypeCalculatedData" returns TRUE, if the image window is an calculated scan image.
Boolean **SetDataTypeCalculatedData** ()	"SetDataTypeCalculatedData" sets the data type to the type : Calculated data.
Boolean **IsScanTypeLineScan** ()	"IsScanTypeLineScan" returns TRUE, if the image window is an line scan image.
Boolean **IsScanTypeMeanOfRoi** ()	"IsScanTypeMeanOfRoi" returns TRUE, if the image window is an 'Mean of ROI' scan image.
Boolean **SetTitle** (BSTR Title)	"SetTitle" sets the name of the image window to the given string 'Title' Returns TRUE if the operation was successful.
Boolean **RecalculateLayout** ()	"RecalculateLayout" recalculate the size of all elements in the image window. Returns TRUE if the operation was successful.
Long **GetNumberEvents** ()	"GetNumberEvents" returns the number of events in the image(Trigger,- Marker- events).
Double **GetEventTime** (long Index)	"GetEventTime" returns the time at which the event with the index 'Index' occurs.
EnumEventType **GetEventType** (long Index)	"GetEventType" returns the type of an event with the index 'Index'. See enums which types exists.
BSTR **GetEventDescription** (long Index)	"GetEventDescription" returns the description of an event with the index 'Index'.
Boolean **AddEvent** (double Time, enumEventType Type, BSTR Description)	"AddEvent" adds an event to the event list. 'Time' (IN) at which time stamp. 'Type' (IN) from which event type. 'Description' (IN) the description

Boolean **RemoveEvent** (long Index)	“RemoveEvent” removes the event at the index ‘Index’ from the event list.
Boolean **IsBusy** ()	
VARIANT **ExtractLine** (long* NumberValues, long Channel, long StartX, long StartY, long StartZ, long StartTime, long EndX, long EndY, long EndZ, long EndTime)	“ExtractLine” extracts image data of a specified region.
Long **SaveToDatabase** (BSTR DatabaseName, BSTR RecordingName)	“SaveToDatabase” stores the image data to an image database with the path ‘DatabaseName’ as recording with the name ‘RecordingName’
DsVectorOverlay* **VectorOverlay** ()	“VectorOverlay” returns a dispatch pointer of the type ‘DsVectorOverlay’.
Boolean **CopySubregion** (long DestChannel, long DestStartX, long DestStartY, long DestStartZ, long DestStartT, long DestStrideX, long DestStrideY, long DestStrideZ, long DestStrideT, DsRecordingDoc* Source, long SrcChannel, long SrcStartX, long SrcStartY, long SrcStartZ, long SrcStartT, long SrcStrideX, long SrcStrideY, long SrcStrideZ, long SrcStrideT, long SizeX, long SizeY, long SizeZ, long SizeT)	“CopySubRegion” copy a region from a given soure image (src-parameters) to this image object to the given position (dest-parameters) Returns TRUE if the function was successful completed.
Boolean **BringToTop** ()	“BringToTop” displays the image as the top level image of all image windows. Returns TRUE if the function was successful completed.
Void **NeverAgainScanToTheImage** ()	“NeverAgainScanToTheImage” sets a flag inside the image window, so that nobody can scan into this image again.

Void **ShowToolbars** (short Show)	“ShowToolbars” displays or hides the toolbars of the image window.
Boolean **Export** (enumExportFormats Format, BSTR FileName, boolean Series, boolean WindowImage, long IndexZ, long IndexT, boolean Mono, long ChannelForRed, long ChannelForGreen, long ChannelForBlue)	“Export” exports the image data into the file ‘FileName’ with the given format ‘Format’. Returns TRUE if the function was successful completed.
Boolean I**sValid** ()	“IsValid” returns TRUE if the image contains valid data.
Void **EnableCrop** (short Enable)	“EnableCrop” enables or disables the ‘Crop-Button’ of the image window toolbar.
Void **EnableReuse** (short Enable)	“EnableReuse” enables or disables the ‘Reuse-Button’ of the image window toolbar.
VARIANT **GetSubregion** (long Channel, long StartX long StartY, long StartZ, long StartT, long StrideX, long StrideY, long StrideZ, long StrideT, long SizeX, long SizeY, long SizeZ, long SizeT, long* BytesPerPixel)	“CopySubRegion” extracts a region from a given soure image (src-parameters). Returns TRUE if the function was successful completed.
long **GetRecordingDoc** ()	Access to the DsRecordingDoc from Interface pointer.

4.10.2. DsRecording

DsRecording	
"DsRecording" Interface to the scan methods and properties.	
Properties	
BSTR **Name**	"Name" is the name of the recording as specified in the "Save" dialog box. The name is used for the title of the recording (image) document after reload from the database. "Name" is initialized to be an empty string.
BSTR **Description**	"Description" is a short description which can be used to quickly select from a small number of recordings in a database. "Description" can be specified by the user in the "Save/Save As" dialog boxes or in the form-view of a database document. "Description" is initialized to be an empty string.
BSTR **Notes**	"Notes" is a fuller description of a Recording. "Notes" can be specified by the user in the "Save/Save As" dialog boxes or in the form-view of a database document. "Notes" is initialized to be an empty string.
Long **SamplesPerLine**	"SamplesPerLine" is the number of samples within each scan line (x-dimension). Note that this is the number of samples used in the scan process. If an image modification (calculation) was done afterwards the scan memory can have a different x-dimension.
Long **LinesPerFrame**	"LinesPerFrame" is the Number of scan lines within each scan plane (y-dimension). Note that this is the number of samples used in the scan process. If an image modification (calculation) was done afterwards the scan memory can have a different y-dimension.
Long **FramesPerStack**	"FramesPerStack" is the number of scan frames within each stack (z-dimension). Note that this is the number of samples used in the scan process. If an image modification (calculation) was done afterwards the scan memory can have a different z-dimension.
Long **StacksPerRecord**	"StacksPerRecord" is the number of stacks within whole record. (time-dimension). Note that this is the number of samples used in the scan process. If an image modification (calculation) was done afterwards the scan memory can have a different time-dimension.
Double **FrameSpacing**	"FrameSpacing" is the center to center distance between adjacent samples on adjacent scan frames (z-direction) in microns. The value was taken at the start of the scan process. After image modifications the z-distance of frames in the scan memory can have a different value.
Double **Rotation**	"Rotation" is the Euler rotation angle of the scan coordinates relative to the microscope coordinates in degrees and is used for rotated scans.
Double **Nutation**	"Notation" is the Euler nutation angle of the scan coordinates relative to the microscope coordinates in degrees, is currently not used, preinitialized to zero and should not be modified.
Double **Precession**	"Precession" is the Euler precession angle of the scan coordinates relative to the microscope coordinates in degrees, is currently not used, preinitialized to zero and should not be modified.

Double **Sample0X**	“Sample0X” is the x-scanner offset in microns. It was used to control the x-offset scanner DAC at scan time.
Double **Sample0Y**	“Sample0Y” is the y-scanner offset in microns. It was used to control the y-offset scanner DAC at scan time.
Double **Sample0Z**	“Sample0Z” is the distance of the scanner focus to the first z image position in microns.
BSTR **Objective**	“Objective” is the name of the objective used at scan time.
Double **ZoomX**	“ZoomX” is the x-scanner zoom factor. It was used to control the x-zoom scanner DAC at scan time. Values are in the range “0.7 ... 8”.
Double **ZoomY**	“ZoomY” is the y-scanner zoom factor. It was used to control the y-zoom scanner DAC at scan time. Values are in the range “0.7 ... 8”.
Double **ZoomZ**	“ZoomZ” is the z-scanner zoom factor. It was used to control the z-zoom scanner DAC at scan time.
BSTR **ScanType**	“ScanType” is currently not used, preinitialized to an empty string and should not be modified.
BSTR **ScanMode**	“ScanMode” is a string that specifies scan mode which can be “Point”, “Line”, “Plane”, “ZScan”, “Stack”, “LineSelect” or “Imported File”. In the latter case it is unknown which scan mode was used.
Boolean **TimeSeries**	“TimeSeries” specifies whether a the scan process was a time series (unequal zero) or not (zero).
BSTR **SpecialScanMode**	“SpecialScanMode” is additional characterization of the scan mode which for the time being specifies which hardware was used for the z-motion. It can be: "NoSpecialMode", "FocusStep", "OnTheFly" or "Zscanner".
Long **ScanDirection**	“ScanDirection” is used to distinguish between uni- and bi-directional scan. “1” means bi-directional, “0” means unidirectional scan.
BSTR **StartScanTriggerIn**	If “StartScanEvent” is 1 (Trigger) “StartScanTriggerIn” specifies which trigger in signal of the user port of the electronic unit should be used for the start of the scan operation. The following trigger signals are supported in version 2.3: “Trigger1” “Trigger2” “Trigger3” “Trigger4”
BSTR **StopScanTriggerIn**	If “StopScanEvent” is 1 (Trigger) “StartScanTriggerIn” specifies which trigger in signal of the user port of the electronic unit should be used for the stop of the scan operation. The following trigger signals are supported in version 2.3: “Trigger1” “Trigger2” “Trigger3” “Trigger4”
BSTR **StartScanTriggerOut**	“StartScanTriggerOut” can be used to enable a trigger out signal at the user port of the electronic unit at start of the scan operation. If no trigger is required an empty string should be used. The following trigger signals are supported in version 2.3: “Trigger1” “Trigger2” “Trigger3” “Trigger4”

BSTR **StopScanTriggerOut**	"StopScanTriggerOut" can be used to enable a trigger out signal at the user port of the electronic unit at stop of the scan operation. If no trigger is required an empty string should be used. The following trigger signals are supported in version 2.0: "Trigger1" "Trigger2" "Trigger3" "Trigger4"
Long **StartScanEvent**	"StartScanEvent" specifies the event that starts the scan operation - 0 - Button (normal operation), - 1 - Trigger slow - Scanner off, PMT high voltage off, reaction 300 msec, - 2 - Trigger normal - Scanner off, PMT high voltage on, reaction 30 msec, - 3 - Trigger fast - Scanner on, PMT high voltage on, reaction 5 msec, - 4 - Start time (see "StartScanTime").
Long **StopScanEvent**	"StopScanEvent" specifies the event that stops the scan operation - 0 - Button (normal operation button), - 1 - Trigger (see "StopScanTriggerIn"), - 2 - End time (see "StopScanTime"). A scan operation is also stopped in all three cases when "SamplesPerLine", "LinesPerPlane", "PlanesPerVolume" and "StacksPerRecord" are reached.
DATE **StartScanTime**	If "StartScanEvent" is equal 2 (start time) "StartScanTime" specifies the date and time when the scan operation should be started.
DATE **StopScanTime**	If "StopScanEvent" is equal 2 (end time) "StopScanTime" specifies the date and time when the scan operation should be stoped.
Boolean **UseROIs**	"UseROIs" specifies whether ROIs should be used (unequal zero) for the scan process or not (zero).
Boolean **FitFramesizeToROIs**	"FitFramesizeToROIs" fits the scan size to the ROIs.
Boolean **NoodleMode**	"NoodleMode" get/set spline scan mode.
Boolean **UseBCCorrection**	"UseBCCorrection" use amplifier gain, amplifier offset, detector gain and AOM interpolation in scan mode z-stack (BC correction).
double **FocusPosABC1**	"FocusPosABC1" set focus position for fererenze 1 for BC correction.
double **FocusPosABC2**	"FocusPosABC2" set focus position for fererenze 2 for BC correction.
long **LineStepNumber**	"LineStepNumber" gets the line step number for y line interpolation.
Methods	
BSTR **SampleData**()	"SampleData" returns a string with a database relative path name of the image file. The String can contain %DatabasePath% which must be replaced by the path of the database path.
Double **SampleSpacing** ()	"SampleSpacing " returns the center to center distance between adjacent samples on the same scan line in microns. The sample spacing can be modified using the zoom factor "ZoomX".

Double **LineSpacing**()	“LineSpacing “ returns the center to center distance between adjacent samples on adjacent scan lines in microns. The sample spacing can be modified using the zoom factor “ZoomY”.
DATE **Sample0Time**()	“Sample0Time” returns the time at which the first sample of the first line (... plane stack) was acquired.
Boolean **IsOrginalScanData**()	“IsOriginalScanData” can be used to obtain the information whether the scan memory contains original scan data or data that was modified afterwards by calculations ...
BSTR **ProcessingSummary**()	“ProcessingSummary” returns a string that contains a short description of the image modifications that were made to the scan memory.
Double **FrameWidth**()	“FrameWidth” returns the scan memory size in x-direction. If an image modification was done this value can be different from the value in “SamplesPerLine“.
Double **FrameHeight**()	“FrameHeight” returns the scan memory size in y-direction. If an image modification was done this value can be different from the value in “LinesPerPlane“.
Boolean **Copy** (DsRecording* Source)	“Copy” copies the scan parameters form a source “DsRecording” object. Only selected entries are copied. The method was designed for remembering and restoring scan parameters.
Double **StackDepth**()	“StackDepth” returns the depth of the scan field (Z-direction) in microns used at scan time. If an image modification was done this value can be different from the depth of the scan memory.
Long **NumberOfChannels**()	“NumberOfChannels” returns the number of channels in the scan memory. If an image modification was done this value can be different from the value returned by “NumberOfChannels”.
Long **TrackCount**()	Get the number of tracks.
Boolean **TrackRemove** (long TrackIndex)	Remove the track at this index .
Long **LaserCount**()	Get the number of lasers.
Boolean **LaserRemove** (long LaserIndex)	Remove a laser at this index.
Boolean **TrackAddNew** (BSTR Name)	Add a track with this name.
Boolean **LaserAddNew** (BSTR Name)	Add a laser with this name.
DsLaser* **LaserObjectByIndex** (long Index, short* Successful)	Get the laser object by index.
DsLaser* **LaserObjectByName** (BSTR Name, short* Successful)	Get the laser object by name.
DsTrack* **TrackObjectByIndex** (long Index, short* Successful)	Get the track object by index.

DsTrack* **TrackObjectByName** (BSTR Name, short* Successful)	Get the track object by name.
DsTrack* **TrackObjectByMultiplexOrder** (long Order, short* Successfull)	Get the track object by multiplexorder (Starts witch 0).
Long **MarkersCount**()	Get the number of markers.
DsMarkers* **MarkersObjectByIndex** (long Index, short* Successful)	Get a markers object by index.
Boolean **MarkersRemove** (long Index)	Remove a marker.
Boolean **MarkersAddNew** (BSTR Name)	Create a new marker.
Long **TimersCount**()	Get the number of timers.
DsTimers* **TimersObjectByIndex** (long Index, short* Successful)	Get a timers object by index.
Boolean **TimersRemove** (long Index)	Remove a timer object.
Boolean **TimersAddNew** (BSTR Name)	Add a timer object a marker.
DsMarkers* **MarkersObjectByName** (BSTR Name, short* Successful)	Get a marker object by name.
DsTimers* **TimersObjectByName** (BSTR Name, short* Successful)	Get a timer object by name.
DsDetectionchannel* **DetectionChannelOfActiveOrder** (long Order, short* Success)	Get the n’st active detectionchannel over all tracks.
Boolean **TrackAddNewBleach** (BSTR Name)	Add a bleachtrack with this name.
DsTrack* **TrackObjectRatio** (short* Success)	Get the ratio track object.
Long **TrackRatioCount**()	Get the number of ratio tracks.
Boolean **TrackRemoveByMultiplexOrder** (long Order)	Remove a track by a given multiplex order.
Long **GetActiveTrackCount** ()	How many tracks are acquired.

BSTR **MakeNewTrackName** ()	Genarate a new track name.
DsTrack* **TrackObjectBleach** (short* Success)	Gets the Bleachtrack object.
void **TimeIntervalActivated** (long Timer)	Signals DS, that the user activated a timer.
boolean **LoadSaveConfiguration** ()	Loads a Recording configuration.
boolean **SaveMarkerSetting** (BSTR SettingName)	Save the marker settings for this recording.
boolean **LoadMarkerSetting** (BSTR SettingName)	Load the marker settings for this recording.
boolean **SaveTimerSetting** (BSTR SettingName)	Save the timer settings for this recording.
boolean **LoadTimerSetting** (BSTR SettingName)	Load the timer settings for this recording.
boolean **SetABC1Reference** ()	Save all Settings for ABC correction Reference 1.
boolean **SetABC2Reference** ()	Save all Settings for ABC correction Reference 2.
boolean **DoZCorrection** ()	Interpolate settings for ABC correction from References for actual focus position.

4.10.3. DsTrack

DsTrack	
"DsTrack" Interface to the methods and properties of a scan track.	
Properties	
BSTR **Name**	"Name" is the name of the track as specified by the user in the "Configuration" window of the user interface program part. The name is used to distinguish different tracks (device settings) in the user interface. The "Name" can be used as selector in the "DsTrack" collection.
Double **TimeBetweenStacks**	For time series "TimeBetweenStacks" contains the time difference between the start of two scan cycles in seconds.
Double **SampleObservationTime**	"SampleObservationTime" is the integration (detection) time for scan samples in microseconds.
Long **SamplingMode**	"SamplingMode" describes which method is used for the scan value generation. 0 = Sample - The values are generated by integration over a time ("SampleObservationTime") without average. 1 = Line-Average - Same as "Sample" but with line average. That means "SamplingNumber" lines are scanned and then averaged. 2 = Frame-Average - not implemented - Same as "Sample" but with frame average. That means "SamplingNumber" frames are scanned and then averaged. 3 = Integration mode- not implemented - The values are generated by integration until a specified threshold is reached. The scan value is generated by the time.
Long **SamplingMethod**	"SamplingMethod" is the method used to calculate the scan values. For the time being we have only calculations for mean values in Sampling mode "1" and "2". 1 = Mean
Long **SamplingNumber**	"SamplingNumber" is the number of samples, lines or planes to process in average modes.
Long **MultiplexType**	"MultiplexType" specifies whether a switch to the next track is done after a Stack - 2, Plane - 1 or Line - 0.
Long **MultiplexOrder**	"MultiplexOrder" is the zero based index for the switch order of the tracks.
BSTR **Collimator1**	**"**Collimator1" is the name of the first collimator in the system.
Long **Collimator1Value**	"CollimatorPosition1" is the position of the first collimator in the system ("IDAvailableCollimator1").
BSTR **Collimator2**	**"**Collimator1" is the name of the second collimator in the system.
Long **Collimator2Value**	"CollimatorPosition2" is the position of the second collimator in the system ("IDAvailableCollimator2").

Boolean **Acquire**	“Acquire” specifies whether the “Track” (setting) should be used (unequal zero) or not (zero) .
Boolean **IsBleachTrack**	“IsBleachTrack” specifies whether the “Track” (setting) specifies bleach parameters (unequal zero) or not (zero).
Boolean **IsBleachAfterScanNumber**	“IsBleachAfterScanNumber” is only used if “IsBleachTrack” and “Acquire“ are unequal zero and specifies whether the bleach action should be done after a specified scan cycle number specified by “BleachScanNumber”. If set to zero, the bleach action is done when “Acquire “ is unequal zero.
Long **BleachScanNumber**	“BleachScanNumber” is the number of the scan cycle where the bleach action should be done and is used only if “IsBleachTrack”, “IsBleachAfterScanNumber “ and “Acquire“ are unequal zero.
BSTR **TriggerIn**	currently not used. If “TriggerIn” is not an empty string the scan controller waits for the specified trigger signal of the user port of the electronic unit before the scan operation for the track is started. The scan controller will wait at ste start of every sacn cycle. The following trigger signals are supported in version 2.3: “Trigger1” “Trigger2” “Trigger3” “Trigger4”.
BSTR **TriggerOut**	currently not used. “TriggerOut” specifies the signal of the user port of the electronic unit which should be used for the trigger out at every start of the track.The following trigger signals are supported in version 2.3: “Trigger1” “Trigger2” “Trigger3” “Trigger4”.
Boolean **IsRatioTrack**	“IsRatioTrack” is a flag that can be used to group all data channels that define oline calculations in a separate track. This flag is not used by the Data Server and the Control Program. That means the OLE client can also use data channels in other tracks to define online operations.
Long **BleachCount**	“BleachCount” is only used if “IsBleachTrack” and “Acquire“ are unequal zero. “BleachCount” is the number of the bleach (scan) cycle a bleach action has to perform.
Boolean **UseBleachParameters**	Get/Set if using the bleach parameters for the next scan.
double **SpiCenterWavelength**	Get/Set the center wavelength for all spi channels.
Methods	
Long **DataChannelCount** ()	Get the number of datachannels.
Boolean **DataChannelRemove** (long Index)	Revome a datachannel by index.
Long **IlluminationChannelCount** ()	Get the number of illuminationchannels.

Boolean **IlluminationChannelRemove** (long Index)	Remove a illumination channel.
Long **DetectionChannelCount** ()	Get the number of detection channels.
Boolean **DetectionChannelRemove** (long Index)	Remove a detectionchannel by index.
Long **BeamSplitterCount** ()	Get the number of beamsplitters.
Boolean **BeamSplitterRemove** (long Index)	Remove a beamsplitter by index.
Boolean **SwitchDetectionChannel** (long Channel, boolean State, long Color, long BitsPerSample)	Switch a detectionchannel on or off.
Boolean **BeamSplitterAddNew** (BSTR name)	Add a new beamsplitter by name.
Boolean **DataChannelAddNew** (BSTR name)	Add a new datachannel by name.
Boolean **DetectionChannelAddNew** (BSTR name)	Add a new detectionchannel by name.
Boolean **IlluminationChannelAddNew** (BSTR name)	Add a new illuminationchannel by name.
Boolean **NumberOfActiveChannels** (long* number)	Get the number of active detection channels.
Boolean **CheckTrack** ()	Check this track against hardwarwe.
Boolean **CheckBeamSplitters** (boolean SplitterNT1)	Check the beamsplitters correct position. “SplitterNT1” means, NT1 was moved
DsDetectionChannel* **DetectionChannelObjectOfActivatedIndex** (long ActiveIndex, short* Successful)	Get the object of the number of switched index.
DsBeamSplitter* **BeamSplitterObjectByChannel** (long Channel, short* Successful)	Get the object by channel.
DsBeamSplitter* **BeamSplitterObjectByIndex** (long Index, short* Successful)	Get the object by index.
DsBeamSplitter* **BeamSplitterObjectByName** (BSTR Name, short* Successful)	Get the object by name.

DsDataChannel* **DataChannelObjectByChannel** (long Channel, short* Successful)	Get the object by channel.
DsDataChannel* **DataChannelObjectByName** (BSTR Name, short* Successful)	Get the object by name.
DsDataChannel* **DataChannelObjectByIndex** (long Index, short* Successful)	Get the object by index.
DsIlluminationChannel* **IlluminationObjectByChannel** (long Channel, short* Successful)	Get the object by channel.
DsIlluminationChannel* **IlluminationObjectByIndex** (long Index, short* Successful)	Get the object by index.
DsIlluminationChannel* **IlluminationObjectByName** (BSTR Name, short* Successful)	Get the object by name.
DsDetectionChannel* **DetectionChannelObjectByChannel** (long Channel, short* Successful)	Get the object by channel.
Boolean **SetAllPinholeDiameter** (double Diameter)	Sets the pinhole diameter for all detection channels.
Boolean **SaveConfigurationSetting** (BSTR ConfigurationName)	Save the settings for a track under a subkey.
Boolean **LoadConfigurationSetting** (BSTR ConfigurationName)	Load the settings for a track from a subkey.
Boolean **LoadSaveConfiguration** ()	Loads or saves a track configuration.
Boolean **CheckNDDBeamSplitters** (short SplitterNDDNT1)	Choose corresponding emission filters for NDD beamsplitters NT1 and NT2.
Boolean **SetDetectionMode** (long ModeNDD)	Set detection mode.
Boolean **GetDetectionMode** (long* Mode)	Get detection mode
double **GetSpiMinCenterWavelength** ()	Get the minimum center wavelength for all spi channels.
double **GetSpiMaxCenterWavelength** ()	Get the maximum center wavelength for all spi channels.
Boolean **SwitchDetectionChannelNoColor** (long Channel, short State)	Acquire a detectionchannel on or off without color.

Boolean **CheckSPIWavelengthRange** ()	Check the wavelength range of all SPI channels within centerwavelength +- bandwidth.

4.10.4. DsDetectionChannel

DsDetectionChannel	
"DsDetectionChannel" Interface to the methods and properties of a detection channel.	
Properties	
BSTR **Name**	The "Name" is used as selector in the "DsDetectionChannel" collection. The following string are used till now: - Ch1 - Ch2 - Ch3 - Ch4 - ChD - Transmission PMT - ChM - Monitor diode
BSTR **PointDetector**	Name of the detector used for the scan data acquisition and must be set to a valid string selector in the "PointDetectors" collection of the LSM Control Program. Depending on the device configuration the following Point detectors can be available: - Pmt1 - Pmt2 - Pmt3 - Pmt4 - PmtD - Transmission PMT - PmtM - Monitor diode
Double **DetectorGainFirstImage**	PMT-voltage of the PMT specified in "IdAvailablePointDetector". For the monitor diode this value has no meaning. If there are more than one planes per stack (z-direction) the PMT voltage is interpolated between "DetectorGainFirstImage" and "DetectorGainLastImage" so that the first plane gets the voltage "DetectorGainFirstImage" and the last plane gets the voltage "DetectorGainLastImage". Note that there is no interpolation in time-direction.
Double **DetectorGainlastImage**	PMT-voltage of the PMT specified in "IdAvailablePointDetector". For the monitor diode this value has no meaning. If there are more than one planes per stack (z-direction) the PMT voltage is interpolated between "DetectorGainFirstImage" and "DetectorGainLastImage" so that the first plane gets the voltage "DetectorGainFirstImage" and the last plane gets the voltage "DetectorGainLastImage". Note that there is no interpolation in time-direction.

BSTR **Amplifier**	The data acquisition electronics has 4 digitizer channels. Every 4 digitizer channel has an amplifier with selectable gain and offset. PMTs and monitor diode must is connected via a multiplexer with these amplifiers. “IdAvailableAmplifier” specifies which amplifier must be used for the PMT (or monitor diode). The following strings should be used: - Amplifier1 - Amplifier2 - Amplifier3 - Amplifier4. In older program versions the strings - DAC211 - DAC212 - DAC213 - DAC214 were used.
Double **AmplifierGainFirstImage**	Amplifier gain voltage of the amplifier specified in “IdAvailableAmplifier”. If there are more than one planes per stack (z-direction) the amplifier gain voltage is interpolated between “AmplifierGainFirstImage” and “AmplifierGainLastImage” so that the first plane gets the voltage “AmplifierGainFirstImage” and the last plane gets the voltage “AmplifierGainLastImage”. Note that there is no interpolation in time-direction.
Double **AmplifierGainLastImage**	Amplifier gain voltage of the amplifier specified in “IdAvailableAmplifier”. If there are more than one planes per stack (z-direction) the amplifier gain voltage is interpolated between “AmplifierGainFirstImage” and “AmplifierGainLastImage” so that the first plane gets the voltage “AmplifierGainFirstImage” and the last plane gets the voltage “AmplifierGainLastImage”. Note that there is no interpolation in time-direction.
Double **AmplifierOffsetFirstImage**	Amplifier offset voltage of the amplifier specified in “IdAvailableAmplifier”. If there are more than one planes per stack (z-direction) the amplifier offset voltage is interpolated between “AmplifierOffsetFirstImage” and “AmplifierOffsetLastImage” so that the first plane gets the voltage “AmplifierOffsetFirstImage” and the last plane gets the voltage “AmplifierOffsetLastImage”. Note that there is no interpolation in time-direction.
Double **AmplifierOffsetLastImage**	Amplifier offset voltage of the amplifier specified in “IdAvailableAmplifier”. If there are more than one planes per stack (z-direction) the amplifier offset voltage is interpolated between “AmplifierOffsetFirstImage” and “AmplifierOffsetLastImage” so that the first plane gets the voltage “AmplifierOffsetFirstImage” and the last plane gets the voltage “AmplifierOffsetLastImage”. Note that there is no interpolation in time-direction.

BSTR **Pinhole**	Name of the pinhole used for the detector. It should be a valid selector in the “Pinholes” collection of the “LSM Control Program”. The following strings are used till now: - PH1 - PH2 - PH3 - PH4 The monitor diode and the transmission PMT have no pinholes one should use an empty string for them.
Double **PinholeDiameter**	Pinhole diameter in microns for the pinhole specified in “Pinhole”.
BSTR **FilterSet1**	Name of the filter set which is located immediately before the detector in the beam path. The following strings are used till now: - EF1 - EF2 - EF3 - EF4 - EF5 (monitor diode) The transmission PMT has no filter set. One should use an empty string for the transmission PMT.
BSTR **Filter1**	Name of the filter which should be used for the filter set “FilterSet1”. It must be a valid filter of the filter collection of the filter set. Use an empty string foe the transmission PMT channel.
BSTR **FilterSet2**	The monitor diode has two filter sets. “FilterSet2**” is the n**ame of the second filter set. The string “EF6” is used till now. One should use an empty string for all other detectors.
BSTR **Filter2**	Name of the filter which should be used for the filter set “FilterSet2”. It must be a valid filter of the filter collection of the filter set. Use an empty string for all other detectors except the monitor diode channel.
BSTR **Integrator**	Every PMT has an integrator. “IDAvailableIntegrator” is the name of the integrator which should be a valid selector of the “Integrators” collection in the “LSM Control Program”.
Double **CountingTrigger**	In photon counting mode (see IntegratorMode) the time until the integrator has reached a threshold is taken as measurement value. CountingTrigger is the threshold voltage in the range 0V..5V.
Long **IntegratorMode**	The integrator can be used in 0 - integration mode or 1 - photon counting mode. In integration mode the measure values are taken by integration over a given time. The time is calculated automatically by the “LSM control program”. In photon counting mode the time until the integrator has reached a threshold is taken as measurement value. The threshold must be specified in “CountingTrigger**”.**
Long **SpecialMode**	Not used till now - set to “0”.
Boolean **Acquire**	With “Acquire” one can disable a detection channel. A value of “0” means disable. A value unequal “0” means “enable”.
Double **AmplifierGain**	Get/Set the amplifier gain for all images.

Double **AmplifierOffset**	Get/Set the amplifier offset for all images
Double **DetectorGain**	Get/Set the detector gain for all images.
Double **DetectorGainABC1**	Get/Set the detector gain ABC1 for all images.
Double **DetectorGainABC2**	Get/Set the detector gain ABC2 for all images.
Double **AmplifierGainABC1**	Get/Set the amplifier gain ABC1 for all images.
Double **AmplifierGainABC2**	Get/Set the amplifier gain ABC2 for all images.
Double **AmplifierOffsetABC1**	Get/Set the amplifier offset ABC1 for all images.
Double **AmplifierOffsetABC2**	Get/Set the amplifier offset ABC2 for all images.
Double **SpiWavelengthStart1**	Get/Set the start wavelength of the first range for SPI.
Double **SpiWavelengthEnd1**	Get/Set the end wavelength of the first range for SPI.
Double **SpiWavelengthStart2**	Get/Set the start wavelength of the second range for SPI.
Double **SpiWavelengthEnd2**	Get/Set the end wavelength of the second range for SPI.
long **SpiSpectralScanChannels**	Get/Set umber of SPI spectral scan channels.
BSTR **DyeName**	Get/Set the name of dye.
BSTR **DyeFolderName**	Get/Set the name of dye folder.
Methods	
Double **CalculateAiryUnits** ()	Calculates the airy unit for this channel.
Double **CalculateZResolution** ()	Calculates the Z resolution for this channel.
Double **CalcPinholeDiameterFromAiry** (double Airy)	Calculates the pinhole diameter for a sertain airy unit.

4.10.5. DsBeamSplitter

DsBeamSplitter	
"DsBeamSplitter" Interface to the settings of a beam splitter object.	
Properties	
BSTR **Filter**	"Filter" is the name of the filter to use in the filter set "FilterSet". It should be a filter specified in the system configuration database for the filter set "FilterSet".
BSTR **FilterSet**	"FilterSet" is the name of the filter set as specified in the system configuration database an used in the "ControlProgram". The following strings are used till now: HT - Main beam splitter NT1 - First dichroic bean splitter NT2 - Second dichroic bean splitter NT3 - Third dichroic bean splitter
BSTR **Name**	The "Name" is used as selector in the "DsBeamSplitter" collection and should be set to the same string as "FilterSet".

4.10.6. DsIlluminationChannel

DsIlluminationChannel	
"DsIlluminationChannel" contains the parameters used to configure of the Acusto Optical Tunable Filters (AOTF) which are responsible for the selection of exitation wavelengths and their intensities.	
Properties	
BSTR **Name**	The "Name" is used as selector in the "DsIlluminationChannel" collection. In older versions the string "AOTF" was used in all Illumination channels (it was a bug). In the newer versions a string with the wavelength in nm is used.
Long **Wavelength**	The wavelength of interest of the light source in nm.
Double **Power**	Power in percent of the AOTF channel or attenuation filter if one.
Boolean **Acquire**	With "Acquire" one can disable a illumination channel. A value of "0" means disable. A value unequal "0" means "enable".
BSTR **DetectionChannelName**	For automatic collimator correction it is required to know which laser line is detected in which point detector. "DetectionChannelName" specifies which detection channel will receive the light of our illumination channel. The string should be a valid name of one existing entry in the "DsDetectionChannel" collection (See "DsDetectionChannel.Name" for possible names) or an empty string to indicate that the connection is unknown. In the latter case no collimator correction is possible.
Double **PowerABC1** ()	Power in percent of the AOTF channel or attenuation filter if one for ABC1 reference.
Double **PowerABC2** ()	Power in percent of the AOTF channel or attenuation filter if one for ABC2 reference.

4.10.7. DsDatachannel

DsDatachannel	
"DsDatachannel" Interface to a data channel. Contains channel dependend parameters of the scan memory and the display and calculation of scan data during the scan operation.	
Properties	
BSTR **Name**	The "Name" is used as selector in the "DsDataChannels" collection. The following string are used till now: - Ch1 - Ch2 - Ch3 - Ch4 - ChD - Transmission PMT - ChM - Monitor diode This string is visible to the user for channel selections in the "LSM Data Server".
Long **MMFIndex**	"MMFIndex" is obsolete., must be –1.
Long **ColorRef**	Is the color which should be used to display the pixel data in the "LSM Data Server" with intensity values in the range 0..255 for the three color components. Msb Lsb 0 blue green red
Long **BitsPerSample**	Specifies the number of bits which should be used to store the pixel data for the data channel. The following values should be used 8 bit channel - value "8" 12 bit channel - value "12".
Long **RatioType**	"RatioType" specifies the type of an online calculation where this "data channel" acts as destination. The folllowing values are possible: 0 - no online calculation - the data channel receives raw scan data. 1 - Online ratio: (S1 + C1) / (S2 + C2) * C3 + C4 2 - Online subtraction: (S1 * C1 + S2 * C3) / C2 + C4 3 - Online ratio: (S1 - S2 * C1) / (S1 + S2 * C2) * C3 + C4 The operands and constanst are (see description of the following methods): S1 - Source1 defined by "RatioTrack1" and "RatioChannel1" S2 - Source2 defined by "RatioTrack2" and "RatioChannel2" C1 - "RatioConstant1" C2 - "RatioConstant2" C3 - "RatioConstant3" C4 - "RatioConstant4"

Long **RatioTrack1**	if “RatioType” is unequal “0” “RatioTrack1“ and “RatioChannel1” identifies the first source operand for an online calculation. “RatioTrack1“ is the multiplex order (see “MultiplexOrder” in DsTrack) of the track of the first operand and “RatioChannel1” is the “Name” property of the data channel of the first source operand in the track “RatioTrack1“. Both source operands must refer to existent data channels for raw data (“RatioType” equal 0).
Long **RatioTrack2**	If “RatioType” is unequal “0” “RatioTrack2“ and “RatioChannel2” identifies the first source operand for an online calculation. “RatioTrack2“ is the multiplex order (see “MultiplexOrder” in DsTrack) of the track of the first operand and “RatioChannel2” is the “Name” property of the data channel of the first source operand in the track “RatioTrack2“. Both source operands must refer to existent data channels for raw data (“RatioType” equal 0).
BSTR **RatioChannel1**	Get/Set the used channel name for source 1
BSTR **RatioChannel2**	Get/Set the used channel name for source 2
Double **RatioConstant1**	If “RatioType” is unequal “0” these properties are the coefficients for online calculations as described in “RatioType**”.**
Double **RatioConstant2**	If “RatioType” is unequal “0” these properties are the coefficients for online calculations as described in “RatioType**”.**
Double **RatioConstant3**	If “RatioType” is unequal “0” these properties are the coefficients for online calculations as described in “RatioType**”.**
Double **RatioConstant4**	If “RatioType” is unequal “0” these properties are the coefficients for online calculations as described in “RatioType**”.**
Boolean **Acquire**	“Acquire” specifies whether the “Data channel ” should be used (unequal zero) or not (zero)

4.10.8. DsLaser

DsLaser	
"DsLaser" Interface to store laser settings used in the last scan process. Contains the information which lasers were used during the scan operation and which laser power was selected (if the laser power is adjustable). These settings are only for information.	
Properties	
BSTR **Name**	Name of the laser used for the scan data acquisition. It must be set to a valid string selector in the "Lasers" collection of the LSM Control Program. Depending on the device configuration the following Lasers can be available: - NeNe1 - NeNe2 - Ar/Kr - Enterprise - Argon - YAG
Double **Power**	Laser power in mW to be used for the scan data acquisition. If the power is not adjustable this number has no meaning.
Boolean **Acquire**	In "Acquire" one can specified whether a laser was used or not during the pixel data acquisition. A modification of "Acquire" does not switch lasers on or off. A value of "0" means the laser was "off". A value unequal "0" the laser was "on".

4.10.9. DsMarkers

DsMarkers	
"DsMarkers" interface to a marker for the actual scanning process. Markers are user defined character strings which are used during a scan operation to record user actions. The marker string is recorded with time and image number if the user activates the maker in the user interface or an event is signaled at the user port of the elektronic unit. "DsMarkers" contains the user settings for that task not the recorded events.	
Properties	
BSTR **Name**	The "Name" is used as selector in the "DsMarkers" collection. The strings "Marker1" to "Marker10" are used in Version 2.3 .
BSTR **Description**	The "Description" is the user specified string which should be recorded with time and image number when the user or an external event has activated the marker.
BSTR **TriggerIn**	"TriggerIn" specifies which input of the user port of the electronic unit triggers the recording of the marker event. If "TriggerInMask" is an empty string we don't look for events at the electronic unit. The following trigger signals are supported in version 2.3: "Trigger1" "Trigger2" "Trigger3" "Trigger4".
BSTR **TriggerOut**	"TriggerOut" specifies which output of the user port of the electronic unit should be set after the user has manually signaled the marker event. An an empty string should be used if no trigger out signal is required. The following trigger signals are supported in version 2.3: "Trigger1" "Trigger2" "Trigger3" "Trigger4".

4.10.10. DsTimers

DsTimers	
"DsTimers" Interface to a timer. Which are used for the scanning process. In time series scans it is possible to change the cycle delay between the end and start of the scanning of stacks. "DsTimers" is member of a collection of timing properties "DsTimers" containing a set of cycle delays. The user can switch between these cycle delays during the scan operation.	
Properties	
Double **TimeInterval**	Scan cycle delay in seconds. It is the time difference between two starts of the scanning of stacks.
BSTR **Name**	The "Name" is used as selector in the "DsTimers" collection. The strings "Timer1" to "Timer6" are used in Version 2.3.
BSTR **Description**	The "Description" is the user specified string which should be recorded with time and image number when the user or an external event has changed the scan cycle delay.
BSTR **TriggerIn**	"TriggerIn" specifies which input of the user port of the electronic unit triggers the activation of the value in "TimeInterval" to be the new scan cycle delay. If "TriggerInMask" is an empty string we don't look for events at the electronic unit. The following trigger signals are supported in version 2.3: "Trigger1" "Trigger2" "Trigger3" "Trigger4".
BSTR **TriggerOut**	"TriggerOut" specifies which output of the user port of the electronic unit should be set after the user has switched the cycle delay. An an empty string should be used if no trigger out signal is required. The following trigger signals are supported in version 2.3: "Trigger1" "Trigger2" "Trigger3" "Trigger4".
Double **ActivationTime**	Time seconds after which the scan cycle delay should be switched to the value specified in "TimeInterval". Set to a value <0 if no activation time is required.
Long **ActivationNumber**	"ActivationNumber" is the number (one based) of the scan cycle after which the scan cycle delay should be switched to the value specified in "TimeInterval". Set to a value <=0 if no activation number is required.

4.11. Additional Objects

4.11.1. DsChannelColors

DsChannelColors	
"DsChannelColors" Interface to the list to select a channel color. "DsChannelColors" is a collection of the colors which the user has generated in the "Channel Colors" dialog of the "LSM Data Server". These colors are used in the channel color selection pop-up windows for selecting channel colors in the "Channels" dialog bar connected to the image windows. Since the user should be able to select channel colors for the data channels in a beam path control of the user interface it is recommended to use this set of user defined channel colors in the beam path control too.	
Methods	
Long **Count** ()	"Count" returns the number of user defined colors in the collection.
Long **Value** (long Index)	With "Value" one can access the color with the zero based index "Index". The returned value is a combination of the three-color components in the range 0…255. Msb Lsb 0 blue green red If there is no entry with the specified index the value "0" (black) is returned.
Void **Value** (long Index, long nNewValue)	With "Value" the color with the zero based index "Index" can be changed. Note that the first five colors (black, red , green , blue and white) cannot be changed. See above for the format of "NewColor". Nothing will happen if there is no entry for the specified index. Check "Count" to determine whether an index exists or not.
Long **AddNew** (long Color)	"AddNew" creates a new entry in the colors collection with the specified color. "AddNew" returns the zero based index of the new color or "-1" in case of an error (insufficient memory).
Boolean **Remove** (long Index)	"Remove" removes the color entry with the specified color in the collection. The entry with indices greater than the specified index are decremented by one. "Remove" returns a value unequal "0" if the entry was removed and a value of "0" if the entry with the specified index could not be found or the index was less than 5.
Void **ColorDialogBox** ()	"ColorDialogBox" calls a modal dialog where the user can create, remove and edit the colors.

4.11.2. DsMruList

DsMruList	
"DsMruList" interface to the MRU list. The List of the last used recording databases. is used to enumerate the full path names of the most recently used image databases. This information can be used to fill the "File" menu items for most recently used files in the main memory of the user interface.	
Properties	
Methods	
Long **Count**()	"Count" returns the number of path names in the collection of most recently used databases.
BSTR **Name** (long Index)	Returns a the full path name of the database file with the specified zero based index "Index". The newest database has the index "0".

4.11.3. DsFluorescenseDatabase

DsFluorescenseDatabase	
"DsFluorescenseDatabase" Interface for access to the Fluorescence database. It contains excitation and emission wavelengths for several dyes.	
Properties	
BSTR **Name**	Name of the dye.
Long **Excitation1From**	Start of the first excitation wavelength range in nm or "0" if there is none.
Long **Excitation1To**	End of the first excitation wavelength range in nm or "0" if there is none.
Long **Excitation2From**	Start of the second excitation wavelength range in nm or "0" if there is none.
Long **Excitation2To**	End of the second excitation wavelength range in nm or "0" if there is none.
Long **Emission1From**	Start of the first emission wavelength range in nm or "0" if there is none.
Long **Emission1To**	End of the first emission wavelength range in nm or "0" if there is none.
Long **Emission2From**	Start of the second emission wavelength range in nm or "0" if there is none.
Long **Emission2To**	End of the second emission wavelength range in nm or "0" if there is none.
Methods	
Long **Count**()	Get the number of database entrys
BOOL **Select** (long Index)	Select a database record
BOOL **AddNew**()	Add a record to the database.
BOOL **Remove** (long Index)	Remove a record from the database.

4.11.4. DsGuidedModeDatabase

DsGuidedModeDatabase	
"DsGuidedModeDatabase" Interface to the guided mode database. Can be used to access the guided mode database. The guided mode database has the same structure as a "normal" image database. The only exception is that there are no image files referenced in the "SampleData" entry of the "Recordings" table because the guided mode is interested in scan parameters only. There is only one guided mode database for all users. The Database is located in the subdirectory "Guided" of the executable file which has called the "LSM Data Server".	
Properties	
DsRecording* **DsRecordingObject**	"DsRecordingObject" returns a dispatch pointer to the "DsRecording" interface which contains the scan parameters. "MoveIndex" or "MoveID" must be called before "Recording" to select a record set in the database.
Long **IDRecording**	"IDRecording" returns the value in the column "IDRecording" of the "Recordings" table for the currently selected row. In case of an error "0" is returned.
BSTR **RecordingName**	Returns the string in the column "RecordingName" of the "Recordings" table for the currently selected row. In case of an error a null-pointer is returned.
BSTR **DatabaseName**	Returns the string with the full path name of the guided mode database. In case of an error a null-pointer is returned.
Methods	
Boolean **MoveIndex** (long Index)	Selects the row in the "Recordings" table of the database specified by the zero based index "Index". "MoveIndex" returns a value equal to "0" in case of an error (the row doesn't exists, ...) and unequal "0" in case of no error.
Boolean **MoveID** (long IdRecording)	Selects the row in the "Recordings" table of the database where the column "IDRecording" has the specified value "IdRecording". "MoveID" returns a value equal to "0" in case of an error (the row doesn't exists, ...) and unequal "0" in case of no error.
Boolean **DeleteRecording** ()	Deletes the current recording (selected with MoveIndex).

4.11.5. DsVectorOverlay

DsVectorOverlay	
"DsVectorOverlay" interface to the overlay drawing procedures.	
Properties	
EnumOverlayDrawingMode **DrawingMode**	**"**DrawingMode" returns actual drawing mode. See enums for all possible modes.
Boolean **OverlayEnabled**	"OverlayEnabled" retruns TRUE if the ovelay drawing is enabled.
Long **LineWidth**	"LineWidth" gets or sets the linewith for the overlay drawing.
Long **Color**	"Color" gets or sets the color for the overlay drawing.
BSTR **FontName**	"FontName" gets or sets the font for the overlay drawing.
Long **FontSize**	"FontSize" gets or sets the font size for the overlay drawing.
Boolean **MeasureMode**	"MeasureMode" gets or sets the MeasureMode for the overlay drawing.
Boolean **ElementEnabled** (long Index)	"ElementEnbled" gets or sets the element at the index 'Index' enabled or not.
Long **ElementLineWidth** (long Index)	"ElementLineWidth" gets or sets the line with of the element at the index 'Index'.
Long **ElementColor** (long Index)	"ElementColor" gets or sets the color of the element at the index 'Index'.
BSTR **ElementFontName** (long Index)	"ElementFontName" gets or sets the font name of the element at the index 'Index'.
Long **ElementFontSize** (long Index)	"ElementFontSize" gets or sets the font size of the element at the index 'Index'.
Boolean **ElementMeasureMode** (long Index)	"ElementMeasureMode" gets or sets the measure mode of the element at the index 'Index'.
Methods	
Long **GetNumberDrawingElements** ()	Get the number of drawing elements in the element list.
Long **GetIndexOfCurrentDrawingElement** ()	"GetIndexOfCurrentDrawingElement" returns the index of the current selected drawing element.
Boolean **SetCurrentDrawingElement** (long Index)	"SetCurrentDrawingElement" sets the drawing element at index 'Index' as current.
Void **RemoveAllDrawingElements** ()	"RemoveAllDrawingElements" removes all drawing elements from the element list.
Void **ShowNumbers** (BOOL Show)	"ShowNumbers" shows or hides the numbers.

Boolean **AddTextDrawingElement** (long Left, long Top, BSTR Text)	“AddTextDrawingElement” adds an text ‘Text’ at the screen position ‘Left’ and ‘Top’ to the element list.
Boolean **AddDrawingElement** (enumOverlayDrawingMode Type, long NumberKnots, VARIANT CoordinatesX, VARIANT CoordinatesY)	“AddDrawingElement” adds a drawing element to the element list.
Boolean **AddSimpleDrawingElement** (enumOverlayDrawingMode Type, long X1, long Y1, long X2, long Y2)	“AddSimpleDrawingElement” adds a drawing element to the element list.
Boolean **GetKnot** (long Index, long Knot, long* CoordinateX, long* CoordinateY)	“GetKnot” returns the coordinates of an knot.
EnumOverlayDrawingMode **GetDrawingElemetType** (long Index)	“GetDrawingElemetType” returns the drawing mode of the element at the index ‘Index’.
void **RemoveDrawingElement** (long Index)	“RemoveDrawingElement” removes the drawing element at the index ‘Index’ from the list.
boolean **MoveDrawingElement** (long Index, long OffsetX, long OffsetY)	“MoveDrawingElement” moves the drawing element at the index ‘Index’ around the offset (X,Y)
VARIANT **MakeRoiMask** (long* Left, long* Top, long* Right, long* Bottom, long BoundaryLeft, long BoundaryTop, long BoundaryRight, long BoundaryBottom, BOOL IncludeOutline)	“MakeRoiMask” creates a ROI mask with the given parameters.
VARIANT **OneElementRoiMask** (long Index, long* Left, long* Top, long* Right, long* Bottom, long BoundaryLeft, long BoundaryTop, long BoundaryRight, long BoundaryBottom, BOOL IncludeOutline)	“OneElementRoiMask” creates a one element mask at the index ‘Index’ in the drawing list with the given parameters.

4.12. Constants

4.12.1 Event enumerations

Name	Number	Description
eEventDsScanStopping	1	DsRecordingDoc will stop scanning.
EEventDsScanStopped	2	DsRecordingDoc has stopped scanning.
EEventDsDocClosing	3	A DsRecordingDoc was closed.
EEventDsRecChanged	4	A DsRecordingDoc was changed.
eEventDsActiveRecChanged	5	The active DsRecording was changed.
eEventDsLineSelectionChanged	6	Line selection changed.
eEventDsRangeSelectionChanged	7	Range selection changed.
eEventDsLineScanLineChanged	8	Line scan line changed
eEventDsMruListChanged	9	Mru list changed
eEventDsCropAreaChanged	10	Crop area changed.
EEventDsStatusDisplay	11	StatusDisplay has changed his visibility.
EEventDsRoiValidState	12	ROI has changed his valid state.
eEventDsBleachRoiValidState	13	Bleach ROI has changed his valid state.
eEventDsScanTimeChanged	14	The scanning time was changed.
eEventDsNoodleModeChanged	15	The noodle mode in the line scan mode was changed.
eEventDsRecordingDocClosing	100	recording doc closing.
eEventDsRecordingClosing	150	recording closing.
eEventDsRecordingDataChanged	151	recording data changed.
eEventDsImageWindowLButtonMouseMoveEvent	201	The user has moved the mouse with a pressed left mouse button over an Image window.
eEventDsImageWindowNoButtonMouseMoveEvent	202	The user has moved the mouse over an Image window with no pressed button.
eEventDsImageWindowLeftButtonDownEvent	203	The user has pressed the left mouse button over an Image window.
eEventDsImageWindowLeftButtonUpEvent	204	The user has released the left mouse button over an Image window.
eEventDsImageWindowDisplaySwitched	205	The image display mode was changed.
eEventDsImageWindowTopLevelWindowChanged	206	The top level image window was changed.
EeventMacroStopped	1000	A macro is stopped.
eEventCalculateOptimalPinholes	1001	Signals the CalculateOptimalPinholes procedure is ready.
EeventDataChanged	1002	Recording Data internally changed.
eEventMacroButtonChanged	1003	Macro Button assignment was changed.
eEventDsImageWindowLeftButtonUpEvent	1004	Left mouse button up in image window.
eEventDsImageWindowDisplaySwitched	1005	Display state of image window has changed.
eEventDsImageWindowTopLevelWindowChanged	1006	Top level image window has changed.
eEventDisplayLaserDlg	1010	Show laser dialog.
eEventDisplayMicroDlg	1011	Show microscope dialog.

eEventDisplayAdminDlg	1012	Show hardware administrator dialog.
eEventDisplayBeampathDlg	1013	Show configuration dialog.
eEventDisplayScanDlg	1014	Show scan dialog.
eEventDisplayTimeDlg	1015	Show time series dialog.
eEventDisplayStageDlg	1016	Show stage dialog.
eEventDisplayBleachDlg	1017	Show beach dialog.
eEventScanStart	1200	Macro Button assignment was changed.
eEventScanStop	1201	Macro Button assignment was changed.

4.12.2 Property Event enumerations

Name	Number	Description
EPropertyEventFiltersets	1	Filterset property changed.
ePropertyEventObjectives	2	ObjectiveRevolver property changed.
ePropertyEventPinholes	3	Pinhole property changed.
ePropertyEventShutters	4	Shutter property changed.
ePropertyEventFocus	5	Focus property changed.
ePropertyEventStage	6	Stage property changed.
ePropertyEventLaser	7	Laser property changed.
ePropertyEventScancontrol	8	Scancontrol property changed.
ePropertyEventLamps	9	Lamp property changed.
ePropertyEventEndOfAutoBC	10	AutoBC End.
ePropertyEventHRZ	11	HRZ property changed.
ePropertyEventTimeInterval	12	TimeInterval property changed.
ePropertyEventScanState	13	ScanState property changed.
eEventUserTrigger	14	User Trigger.
eEventAutoDdsOk	15	Auto Dds succeeded.
eEventAutoDdsFailedIntensity	16	Auto Dds failed intensity.
eEventAutoDdsFailedRange	17	Auto Dds failed range.
eEventDspStop	18	Auto Dds failed stoped.
eEventState	19	DSP stoped.
eEventPotentialObjectives	20	Potential Objectives changed.
eEventBootProgress	21	Boot state changed.
eEventLaserline	22	Laser lines property changed.
eEventBootAbort	23	Boot canceld.
eEventScanHalt	24	Scan canceld.
ePropertyEventLsm5Vba	33	Lsm5Vba property changed.

4.12.3 enumMultiplexType enumeration

Name	Number	Description
eMultiplexTypeLine	0	Next track after a line.
eMultiplexTypeFrame	1	Next track after a frame.
eMultiplexTypeStack	2	Not supported. Next track after a Stack.

4.12.4 enumKind enumeration

Name	Number	Description
eKindManual	0	Hardware object is only manual.
eKindCoded	1	Hardware object is coded.
eKindMotorized	2	Hardware object is coded and motorized.
eKindFixed	3	Object is fixed.

4.12.5 enumStartScan enumeration

Name	Number	Description
eStartScanUser	0	Timeseries scan start by user action.
eStartScanTrigger	3	Timeseries scan start by trigger.
eStartScanTime	4	Timeseries scan start by timer.

4.12.6 enumStopScan enumeration

Name	Number	Description
eStopScanUser	0	Scan stopped by user.
eStopScanTrigger	1	Scan stopped by trigger.
eStopScanTime	2	Scan stopped by timer.
eStopScanNumber	3	Scan stopped by number.

4.12.7 enumSamplingMode enumeration

Name	Number	Description
ePixelAverage	0	Average over pixels.
eLineAverage	1	Average over lines.
eFrameAverage	2	Average over frames.
eIntegration	3	Integration.

4.12.8 enumSamplingMethod enumeration

Name	Number	Description
eMeanAverage	1	Mean Average.
eSumAverage	2	Sum Average.
eMaxAverage	3	Max Average.
eMinAverage	4	Min Average.
eMedianAverage	5	Median Average.

4.12.9 enumScanDirection enumeration

Name	Number	Description
eSingleForeward	0	Single Forward scan direction
eBidirectional	1	Bidirectional scan direction

4.12.10 enumChannelNumber enumeration

Name	Number	Description
eChannelSPI1	1	SPI Channel number 1
eChannelSPI2	2	SPI Channel number 2
eChannelSPI3	3	SPI Channel number 3
eChannelSPI4	4	SPI Channel number 4
eChannelSPI5	5	SPI Channel number 5
eChannelSPI6	5	SPI Channel number 6
eChannelSPI7	7	SPI Channel number 7
eChannelSPI8	8	SPI Channel number 8
eChannelSPISpectra	9	SPI Channel for spectral scan
eChannel1	10	Channel number 1
eChannel2	11	Channel number 2
eChannel3	12	Channel number 3
eChannel4	13	Channel number 4
eChannel5	14	Channel number 5
eChannel6	15	Channel number 6
eChannel7	16	Channel number 7
eChannel8	17	Channel number 8
eMonitorChannel	18	MonitorChannel
eTransmissionChannel	19	TransmissionChannel
eChannelNDD2	20	NDD Channel number 2
eChannelNDD3	21	NDD Channel number 3
eChannelNDD4	22	NDD Channel number 4

4.12.11 enumShutterState enumeration

Name	Number	Description
eShutterErrorPos	0	Shutter on Error position
eShutterVisPos	1	Shutter on VIS position
eShutterTVPos	2	Shutter on TV position
eShutterLsmPos	3	Shutter on LSM position

4.12.12 enumVertShutterState enumeration

Name	Number	Description
eVertShutterErrorPos	0	VertShutter on Error position
eVertShutterInPos	1	VertShutter on the pushed position
eVertShutterOutPos	2	VertShutter on the out position

4.12.13 enumShutterState enumeration

Name	Number	Description
eSystemErrorPos	0	System on Error position
eSystemLsmPos	1	System on Lsm position
eSystemFcsPos	2	System on FCS position
eSystemVisPos	3	System on Vis position
eSystemTVPos	4	System on TV position

4.12.14 enumScanMode enumeration

Name	Number	Description
eScanModePoint	0	Scan a Point
eScanModeLine	1	Scan a Line
eScanModeLineSelect	2	Make a lineselect scan.
eScanModeFrame	3	Scan a Frame
eScanModeStack	4	Scan a Stack
eScanModeZScan	5	scan a line in z direction

4.12.15 enumSpecialScanMode enumeration

Name	Number	Description
eNoSpecialScanMode	0	No special Scanmode for z scan.
eSpecialScanModeFocusStep	1	ZScan with the normal Focus Step.
eSpecialScanModeOnTheFly	2	ZScan without stop between scans.
eSpecialScanModeZScanner	3	Special Scanmode Z Scanner
eSpecialScanModeHrzStep	4	Use the HRZ for Z positioning.

4.12.16 enumLaserState enumeration

Name	Number	Description
eLaserOff	0	The Laser is off.
eLaserOn	1	The laser is on.
eLaserStandby	2	The laser is in the standby mode.

4.12.17 enumScanState enumeration

Name	Number	Description
eReady	0	Ready
eScanning	1	Scanning
eBleaching	2	Bleaching
eWaitingForTrigger	3	waiting for trigger
eWaitingForStartTime	4	waiting for start time
ePause	5	Paused
eWaitInterval	6	waiting for interval end

4.12.18 enumDisplayMode enumeration

Name	Number	Description
eDisplayModeNone	0	No display mode
eDisplayModeXY	1	Normal XY image mode
eDisplayModeSplitXY	2	Split display mode.
eDisplayModeOrthoSections	3	Display image as orthogonal sections.
eDisplayModeCut	4	Displays the cut mode.
eDisplayModeGallery	5	Displays a stack in gallery mode.
eDisplayModePseudo3d	6	The pseudo 3D visualization of an XYZ image is active.
eDisplayModeHistogram	7	Displays the histogram presentation.
eDisplayModeProfile	8	Displays the profile of an XYZ image
eDisplayModeColocalisation	9	Displays the colocalisation .
eDisplayModeMeanOfRoi	10	Displays the image data as Main of ROI .
eDisplayModeTopography	11	Topography mode of xyz images.
eDisplayModePrintPreview	12	The image is in the print preview mode.
eDisplayModeLinescan	13	Displays the image in line mode.
eDisplayModeLineSelect	14	Line select is active
eDisplayModeXT	15	XT presentation of a XT scan
eDisplayModeTX	16	TX presentation of a XT scan
eDisplayModeSplitXT	17	Split XT presentation of a XT scan
eDisplayModeSplitTX	18	Split XT presentation of a XT scan
eDisplayModeIntensityProfile	19	Profile display is active.
eDisplayModeMeanOfLine	20	Display is in the mean of line mode.
eDisplayModeScanMeanOfRoi	21	Display is in the mean of roi mode.

4.12.19 enumDataserverEvents enumeration

Name	Number	Description
eRootScanStopping	1	DsRecordingDoc will stop scanning.
eRootScanStopped	2	DsRecordingDoc has stopped scanning.
eRootClosing	3	A DsRecordingDoc was closed.
eRootReuse	5	The active DsRecording was changed.
eRootLineSelectionChanged	6	Line selection changed.
eRootRangeSelectionChanged	7	Range selection changed.
eRootLinescanLineChanged	8	Line scan line changed
eRootRecentFilelistChanged	9	Mru list changed
eRootCropAreaChanged	10	Crop area changed.
eRootStatusDialogVisibiltyChanged	11	StatusDisplay has changed his visibility.
eRootRoisListValidStateChanged	12	ROI has changed his valid state.
eRootBleachRoiListValidStateChanged	13	Bleach ROI has changed his valid state.
eRootScanTimeChanged	14	The scanning time was changed.
eRootNoodleModeChanged	15	The noodle mode in the line scan mode was changed.
eImageWindowLButtonMouseMoveEvent	201	The user has moved the mouse with a pressed left mouse button over an Image window.
eImageWindowNoButtonMouseMoveEvent	202	The user has moved the mouse over an Image window with no pressed button.
eImageWindowLeftButtonDownEvent	203	The user has pressed the left mouse button over an Image window.
eImageWindowLeftButtonUpEvent	204	The user has released the left mouse button over an Image window.
eImageWindowDisplaySwitched	205	The image display mode was changed.
eImageWindowTopLevelWindowChanged	206	The top level image window was changed.

4.12.20 enumEventType enumeration

Specify the event types in timeseries.

Name	Number	Description
eEventTypeNone	0	
eEventTypeMarker	1	A marker was set
eEventTypeTimerChange	2	The scan interval was changed
eEventTypeBleachStart	3	The bleach procedure was started
eEventTypeBleachStop	4	The bleach procedure is stopped.
eEventTypeTrigger	5	A Trigger was set.

4.12.21 enumOverlayDrawingMode enumeration

Defines the type of an overlay drawing element.

Name	Number	Description
eDrawingModeNone	0	
eDrawingModeSelect	1	Element is a Selection.
eDrawingModeLine	2	Element is a Line.
eDrawingModeScaleBar	3	Element is a scale bar
eDrawingModeRectangle	4	Element is a rectangle
eDrawingModeEllipse	5	Element is a ellipse
eDrawingModeCircle	6	Element is a circle
eDrawingModeOpenPolyLine	7	Element is a open polygon curve
eDrawingModeClosedPolyLine	8	Element is a closed polygon curve
eDrawingModeOpenBezier	9	Element is a open bezier curve.
eDrawingModeClosedBezier	10	Element is a closed bezier curve
eDrawingModeOpenArrow	11	Element is a open arrow
eDrawingModeClosedArrow	12	Element is a closed arrow
eDrawingModeText	13	Element is text

4.12.22 enumExportFormats enumeration

Defines the format in which the image will be exported.

Name	Number	Description
eExportNone	0	No format
eExportBmp	1	Bitmap format
eExportJpegPoor	2	JPEG format, high compression rate.
eExportJpegMedium	3	JPEG format, medium compression rate.
eExportJpegAccurate	4	JPEG format, low compression rate.
eExportGif	5	GIF format
eExportTiff	6	TIFF format, 8 bit , no compression
eExportTiffLzw	7	TIFF format, 8 bit, Lzw compression.
eExportTiff12Bit	8	TIFF format, 12 bit, no compression
eExportTiff16Bit	9	TIFF format, 16 bit , no compression
eExportLsm4Planar	10	LSM4 Planar format.
eExportLsm4Chunky	11	LSM4 Chunky format.

4.12.23 enumPowerChangeabilty enumeration

Defines changability of laser power, combination of several flags is possible

Name	Number	Description
ePower_changeable_no	0	Not changable
ePower_changeable_percent	1	Changeable in percent
ePower_changeable_mWatt	2	Changeable in watt
ePump_Power_accessible	4	Pump power changeable in percent
ePump_Power_changeable_mWatt	8	Pump power changeable in watt
eBandwidth_accessible	16	Bandwidth changable
eStandby_possible_no	23	Standby mode possible

4.12.24 eDetectorTypeCode enumeration

Name	Number	Description
eTypeInt	1	Internal detector
eTypeMonitor	2	Monitor diod
eTypeTransmission	3	Transmission detector
eTypeExt	4	External detector
eTypeNDD	5	NDD detector
eTypeSPI	6	SPI detector

4.12.25 eZSelectionMode enumeration

Name	Number	Description
eZSelectZSectioning	0	z sectioning
eZSelectFirstLast	1	eZSelectFirstLast
eZSelectHRZ	2	eZSelectHRZ

4.12.26 enumTypeCode enumeration

Name	Number	Description
eTypeAOTF	0	AOTF
eTypeFilterwheel	1	Filter wheel
eTypeServo	2	servo

4.12.27 enumMessageType enumeration

Name	Number	Description
eTypeInfo	0	info
eTypeError	1	error
eTypeYesNo	2	question
eTypeOkCancel	3	question

4.12.28 enumDetectionMode enumeration

Name	Number	Description
eDetectionModeSPI	0	SPI only
eDetectionModeAdvanced	1	SPI and conventionel confocal
eDetectionModeNDD	2	Non descanned detection
eDetectionModeSpectra	3	SPI spectral scan