8.11. Component AimFcsController

IAimFcsAcquisitionParameters

The "IAimFcsAcquisitionParameters" interface is one of four interfaces for acquisition parameters. Via this interface the parameter which define the measurement protocol with channel, repetition and timing properties are accessible.

Methods

HRESULT **get_Name** ([out,retval] BSTR* ppName)
HRESULT **put_Name** ([in] BSTR pName)

The "Name" property specifies the name for the set of acquisition parameters which has been choosen by the user.

ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the currently used name.
pName	[in]	A pointer to a character string with the name for the set of acquisition parameters
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Comment** ([out,retval] BSTR* ppComment)
HRESULT **put_Comment** ([in] BSTR pComment)

The "Comment" property is a user specified text with additional informations about the acquisition parameter set.

ppComment	[out]	A pointer to a variable which will receive the pointer to the character string with the acquisition parameter set comment.
pComment	[in]	A pointer to a character string with the acquisition parameter set comment.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Category** ([out,retval] BSTR* ppCategory)
HRESULT **put_Category** ([in] BSTR pCategory)

The "Category" property is an additional text which can be used by the user to categorize acquisition parameter sets.

ppCategory	[out]	A pointer to a variable which will receive the pointer to the character string with the acquisition parameter set categorization text.
pCategory	[in]	A pointer to a character string with the acquisition parameter categorization text.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Status** ([out,retval] BSTR* ppStatus)
HRESULT **put_Status** ([in] BSTR pStatus)

The "Status" property is an additional text with informations about design state of the acquisition parameter set specified by the user.

ppStatus	[out]	A pointer to a variable which will receive the pointer to the character string with the design state of the acquisition parameter set
pStatus	[in]	A pointer to a character string with the design state of the acquisition parameter set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_User** ([out,retval] BSTR* ppUser)
HRESULT **put_User** ([in] BSTR pUser)

The "User" property specifies the login name of user who created the acquisition parameter set.

ppUser	[out]	A pointer to a variable which will receive the pointer to the character string with the name of user.
pUser	[in]	A pointer to a character string with the name of the name of user.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DateTime** ([out,retval] double* pdTime)
HRESULT **put_DateTime** ([in] double dTime)

The "DateTime" property specifies the date and time of the last modification of the aquisition parameter set.

pdTime	[out]	A pointer to a variable which will receive the time of the last modification of the aquisition parameter set.
dTime	[in]	The time of the last modification of the aquisition parameter set in days relative to December 30, 1899 0:00 a.m. The fractional part of the value represents the time of day.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_ChannelEnabled ( [in] long                        lChannel  ,
                             [out, retval] VARIANT_BOOL*      pbEnabled )
HRESULT put_ChannelEnabled ( [in] long                        lChannel  ,
                             [in] VARIANT_BOOL                bEnabled  )
```

The "ChannelEnabled" property specifies if an detection channel of the correlator is used during data acqisition.

lChannel	[in]	The zero based index of the correlator channel. This is the same index as used for the correlator channel methods of the hardware information interface.
pbEnabled	[out]	A pointer to a variable which will receive VARIANT_TRUE if the channel is used and VARIANT_FALSE if not.
bEnabled	[in]	A value different from zero switches the correlator channel on. The value zero sitches the correlator channel off.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_ChannelName ( [in] long                lChannel  ,
                          [out,retval] BSTR*       ppName    )
HRESULT put_ChannelName ( [in] long                lChannel  ,
                          [in] BSTR                pName     )
```

The "ChannelName" property specifies a user specified name for a data channel which has been used during acquistion of the FCS data set.

lChannel	[in]	The zero based index of the channel.
ppName	[out]	A pointer to a variable which will receive the user specified channel name.
pName	[in]	A pointer to a character string with the user specified channel name.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_BleachTime ( [out, retval] double*  pdBleachTime )
HRESULT put_BleachTime ( [in] double            dBleachTime  )
```

The "BleachTime" property specifies the time the sample is illuminated prior the data acquisition. This bleaching is done for each sample position.

pdBleachTime	[out]	A pointer to a variable which will receive the currently used sample bleach time in seconds.
dBleachTime	[in]	The new sample bleach time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MeasurementTime** ([out, retval] double* pdMeasurementTime)
HRESULT **put_MeasurementTime** ([in] double dMeasurementTime)

The "MeasurementTime" property specifies the duration of the data acquisiton per sample position, repeat index and kinetics index.

pdMeasurementTime	[out]	A pointer to a variable which will receive the currently used duration of the acqisition in seconds.
dMeasurementTime	[in]	The new duration of the acqisition in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MeasurementRepeat** ([out, retval] long* plMeasurementRepeat)
HRESULT **put_MeasurementRepeat** ([in] long lMeasurementRepeat)

The "MeasurementRepeat" property specifies the number of repetitions for the data acquisition at one sample position and kinetics index.

plMeasurementRepeat	[out]	A pointer to a variable which will receive the currently used number of repetitions.
lMeasurementRepeat	[in]	The new number of repetitions.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Kinetics** ([out, retval] VARIANT_BOOL* pbKinetics)
HRESULT **put_Kinetics** ([in] VARIANT_BOOL bKinetics)

The "Kinetics" property specifies if the multiple acquistions are performed at one sample position with delay for kinetic analysis. The "KineticsCount" and "KineticsMeasurementStart" properties are not used when "Kinetics" is off.

pbKinetics	[out]	A pointer to a variable which will receive "VARIANT_TRUE" if the kinetics measurement is enabled and "VARIANT_FALSE" if not.
bKinetics	[in]	The new state of the kinetcs flag.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_KineticsCount** ([out, retval] long* plKineticsCount)
HRESULT **put_KineticsCount** ([in] long lKineticsCount)

The "KineticsCount" property specifies the number of acquistions at one sample position with delay for kinetic analysis.

plKineticsCount	[out]	A pointer to a variable which will receive the currently used number of aqcusitions.

lKineticsCount	[in]	The new number of aqcusitions.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_KineticsMeasurementStart** ([in] long lKineticIndex ,
[out, retval] double* pdStartTime)
HRESULT **put_KineticsMeasurementStart** ([in] long lKineticIndex ,
[in] double dStartTime)

The "KineticsMeasurementStart" property specifies the time of the start of the acquisition of the first data of an kinetics measurement relative to the end of the bleach time.

lKineticIndex	[in]	The zero based kinetic measurement index.
pdStartTime	[out]	A pointer to a variable which will receive the currently used acquistion start time.
dStartTime	[in]	The new acquistion start time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SequentialIllumination**
([out, retval] VARIANT_BOOL* pbSequentialIllumination)
HRESULT **put_SequentialIllumination**
([in] VARIANT_BOOL bSequentialIllumination)

The "SequentialIllumination" property specifies if multi-channel acquisition is performed with sequential illumination to avoid blead through.

pbSequentialIllumination	[out]	A pointer to a variable which will receive "VARIANT_TRUE" if the data are acquired in sequential mode and "VARIANT_FALSE" if the data are acquired in simultaneous mode.
bSequentialIllumination	[in]	The value 0 indicates that the data for different illuminations are acquitred simultaneously. A different value indicates that they are acquired in sequential mode.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SequentialIlluminationPeriod**
([out, retval] double* pdSequentialIlluminationPeriod)
HRESULT **put_SequentialIlluminationPeriod**
([in] double dSequentialIlluminationPeriod)

The "SequentialIlluminationPeriod" property specifies illumination period at one scanner position

in "eFcsSamplePositionModeSequential" mode.

pdSequentialIlluminationPeriod	[out]	A pointer to a variable which will receive the currently used illumination period in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SpotIlluminationPeriod** ([out, retval] double* pdSpotIlluminationPeriod)
HRESULT **put_SpotIlluminationPeriod** ([in] double dSpotIlluminationPeriod)

The "SpotIlluminationPeriod" property specifies illumination period at one scanner position in "eFcsSamplePositionModeSequential" mode.

pdSpotIlluminationPeriod	[out]	A pointer to a variable which will receive the currently used illumination period in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **EstimateMeasurementTime** ([in] long lKineticIndexCount ,
[out] double* pdTime)

The "EstimateMeasurementTime" method calculates the estimated end time for the acquisitions of the specified kinetic index relative to the start of the measurement.

lKineticIndexCount	[in]	The zero based last kinetic index to consider.
pdTime	[out]	A pointer to a varibale which will receive the estimated measurement time.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Dump** ([out,retval] BSTR* pState)

The "Dump" method generates a string with the informations about the current state of the acquisition parameters.

pState	[in]	A pointer to a variable which will receive the string with the acquisition parameter state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsBeamPathParameters

The "IAimFcsBeamPathParameters" interface is one of four interfaces for acquisition parameters. Via this interface the parameters for the positions of beam path hardware components during acquisition are accessible.

Methods

HRESULT **get_Name** ([out,retval] BSTR* ppName)
HRESULT **put_Name** ([in] BSTR pName)

The "Name" property specifies the name of the beam path which has been choosen by the user.

ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the beam path.
pName	[in]	A pointer to a character string with the name of the beam path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_FilterSetPosition** ([in] long lFilterSet ,
[out,retval] long* plPosition)
HRESULT **put_FilterSetPosition** ([in] long lFilterSet ,
[in] long lPosition)

The "FilterSetPosition" property specifies the index of the filter in the filter set specified by index that is used during acquisition. A negative index indicates that the filter set is not controlled by the FCS controller.

lFilterSet	[in]	The zero based index of the filter set. This is the same index as used for the filter set methods of the hardware information interface.
plPosition	[out]	A pointer to a variable which will receive the zero based index of the currently used filter.
lPosition	[in]	The zero based index of the new filter to use.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PinholeDiameter** ([in] long lPinhole ,
[out,retval] double* pdDiameter)
HRESULT **put_PinholeDiameter** ([in] long lPinhole ,
[in] double dDiameter)

The "PinholeDiameter" property specifies the diameter of the pinhole specified by pinhole index

that is used during acquisition.

lPinhole	[in]	The zero based index of the pinhole. This is the same index as used for the pinhole methods of the hardware information interface.
pdDiameter	[out]	A pointer to a variable which will receive the currently used pinhole diameter.
dDiameter	[in]	The new pinhole diameter in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DetectorVoltage** ([in] long lDetector ,
[out,retval] double* pdVoltage)
HRESULT **put_DetectorVoltage** ([in] long lDetector ,
[in] double dVoltage)

The "DetectorVoltage" property specifies the high voltage of an GaAsP detector. The "GetDetectorMinimumVoltage" and "GetDetectorMaximumVoltage" methods of the hardware information interface can be used to determine if the high voltage is controllable. Thei is the case when the minimum and maximum values are different.

lDetector	[in]	The zero based index of the detector. This is the same index as used for the detector methods of the hardware information interface.
pdVoltage	[out]	A pointer to a variable which will receive the currently used voltage.
dVoltage	[in]	The voltage for the detector.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AttenuatorPower** ([in] long lAttenuator ,
[out,retval] double* pdPower)
HRESULT **put_AttenuatorPower** ([in] long lAttenuator ,
[in] double dPower)

The "AttenuatorPower" property specifies the relative power of the light that transmits the attenuator.

lAttenuator	[in]	The zero based index of the attenuator. This is the same index as used for the attenuator methods of the hardware information interface.
pdPower	[out]	A pointer to a variable which will receive the currently used relative power in the range 0..1.
dPower	[in]	The new relative power in the range 0..1.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AttenuatorOn** ([in] long lAttenuator ,
[out,retval] VARIANT_BOOL* pbOn)
HRESULT **put_AttenuatorOn** ([in] long lAttenuator ,

[in] VARIANT_BOOL bOn)

The "AttenuatorOn" property specifies if the attenuator is on or off during the data acquistion.

lAttenuator	[in]	The zero based index of the attenuator. This is the same index as used for the attenuator methods of the hardware information interface.
pbOn	[out]	A pointer to a variable which will receive VARIANT_TRUE if the attenuator is used and VARIANT_FALSE if not.
bOn	[in]	A value different from zero switches the attenuator on during acquisition. The value zero switches the attenuator off during acquisition.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_BleachAttenuatorPower** ([in] long lAttenuator ,
[out,retval] double* pdPower)
HRESULT **put_BleachAttenuatorPower** ([in] long lAttenuator ,
[in] double dPower)

The "BleachAttenuatorPower" property specifies the relative power of the light that transmits the attenuator during the pre-bleach period.

lAttenuator	[in]	The zero based index of the attenuator. This is the same index as used for the attenuator methods of the hardware information interface.
pdPower	[out]	A pointer to a variable which will receive the currently used relative power in the range 0..1.
dPower	[in]	The new relative power in the range 0..1.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_BleachAttenuatorOn** ([in] long lAttenuator ,
[out,retval] VARIANT_BOOL* pbOn)
HRESULT **put_BleachAttenuatorOn** ([in] long lAttenuator ,
[in] VARIANT_BOOL bOn)

The "BleachAttenuatorOn" property specifies if the attenuator is on or off during the pre-bleach period.

lAttenuator	[in]	The zero based index of the attenuator. This is the same index as used for the attenuator methods of the hardware information interface.
pbOn	[out]	A pointer to a variable which will receive VARIANT_TRUE if the attenuator is used and VARIANT_FALSE if not.
bOn	[in]	A value different from zero switches the attenuator on during acquisition. The value zero switches the attenuator off during acquisition.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Dump** ([out,retval] BSTR* pState)

The "Dump" method generates a string with the informations about the current state of the acquisition parameters.

pState	[in]	A pointer to a variable which will receive the string with the acquisition parameter state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsCalibration

The "IAimFcsCalibration" interface is the interface for access to FCS acquisition calibration data and for calibration routines.

Methods

HRESULT **get_ListPositionOffsetX** ([out, retval] double* pdOffsetX)
HRESULT **put_ListPositionOffsetX** ([in] double dOffsetX)

The "ListPositionOffsetX" property specifies the offsets of the x-coordinate of the laser focus position in the FCS microscope mode relative to the laser focus position LSM microscope mode. This offset value is subtracted form the x-coodinate of a measurement position in the position list dtermined in an LSM image to result in the position that is used for FCS acquisition.

pdOffsetX	[out]	A pointer to a variable which will receive the currently used offset.
dOffsetX	[in]	The new offset in metre.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ListPositionOffsetY** ([out, retval] double* pdOffsetY)
HRESULT **put_ListPositionOffsetY** ([in] double dOffsetY)

The "ListPositionOffsetY" property specifies the offsets of the y-coordinate of the laser focus position in the FCS microscope mode relative to the laser focus position LSM microscope mode. This offset value is subtracted form the y-coodinate of a measurement position in the position list dtermined in an LSM image to result in the position that is used for FCS acquisition.

pdOffsetY	[out]	A pointer to a variable which will receive the currently used offset.
dOffsetY	[in]	The new offset in metre.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ListPositionOffsetZ** ([out, retval] double* pdOffsetZ)
HRESULT **put_ListPositionOffsetZ** ([in] double dOffsetZ)

The "ListPositionOffsetZ" property specifies the offsets of the z-coordinate of the laser focus position in the FCS microscope mode relative to the laser focus position LSM microscope mode. This offset value is subtracted form the z-coodinate of a measurement position in the position list dtermined in an LSM image to result in the position that is used for FCS acquisition.

pdOffsetZ	[out]	A pointer to a variable which will receive the currently used offset.

dOffsetZ	[in]	The new offset in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **StartPinholeAdjustment** (double dDiameter ,
double dStartX ,
double dStartY ,
double dStartZ ,
double dEndX ,
double dEndY ,
double dEndZ ,
long lPositions)

The "StartPinholeAdjustment" method starts the data acquisition for a pinhole adjustment.

dDiameter	[in]	The pinhole diameter to use in metre.
dStartZ	[in]	The pinhole x-, y- and z- servo position for the acquisition of the first count rate result.
dEndZ	[in]	The pinhole x-, y- and z- servo position for the acquisition of the last count rate result. The positions for the other results are calaulated by linear interpolation.
lPositions	[in]	The number of acquisition results.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeAdjustmentResultArraySize** ([out, retval] long* plSize)

The "GetPinholeAdjustmentResultArraySize" method can be used to determine the number of count rate value acquired so far for a data acquisition for pinhole adjustment.

plSize	[out]	A pointer to a variable which will receive the number of acquired count rate values.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeAdjustmentResult** ([in] long lPositions ,
[out, size_is(Positions)] double* pdResult)

The "GetPinholeAdjustmentResult" method copies the specified number of count rate value to the provided buffer.

lPositions	[in]	The number of count rate values to copy.
pdResult	[out]	A pointer to the array which will receive receive the acquired count rates.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_PinholePositionX ( [in] long              lPinhole ,
                               [out,retval] double*   pdPositionX )
HRESULT put_PinholePositionX ( [in] long              lPinhole ,
                               [in] double            dPositionX )
```

The "PinholePositionX" property specifies the position of the x-servo of the pinhole specified by pinhole. When the position is set the specified position is written to the calibration file.

lPinhole	[in]	The zero based index of the pinhole. This is the same index as used for the pinhole methods of the hardware information interface.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_PinholePositionY ( [in] long              lPinhole ,
                               [out,retval] double*   pdPositionY )
HRESULT put_PinholePositionY ( [in] long              lPinhole ,
                               [in] double            dPositionY )
```

The "PinholePositionY" property specifies the position of the y-servo of the pinhole specified by pinhole. When the position is set the specified position is written to the calibration file.

lPinhole	[in]	The zero based index of the pinhole. This is the same index as used for the pinhole methods of the hardware information interface.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_PinholePositionZ ( [in] long              lPinhole ,
                               [out,retval] double*   pdPositionZ )
HRESULT put_PinholePositionZ ( [in] long              lPinhole ,
                               [in] double            dPositionZ )
```

The "PinholePositionZ" property specifies the position of the z-servo of the pinhole specified by pinhole. When the position is set the specified position is written to the calibration file.

lPinhole	[in]	The zero based index of the pinhole. This is the same index as used for the pinhole methods of the hardware information interface.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT GetPinholeAdjustmentDate ( [in] long      lPinhole ,
                                   [out] double*  pdDate )
```

The "GetPinholeAdjustmentDate" method determines the date and time where a pinhole has been adjusted for the current positions of the filter sets.

lPinhole	[in]	The zero based index of the pinhole. This is the same index as used for the

pinhole methods of the hardware information interface.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeAiryUnits** ([in] long lPinhole ,
[out] double* pdAiryUnits)

The "GetPinholeAiryUnits" method can calculates the airy units for the specified pinhole.

lPinhole	[in]	The zero based index of the pinhole in the pinholes list of the hardware information interface.
pdAiryUnits	[out]	A pointer to the variable which will receive the airy units.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CollimatorPosition** ([in] long lCollimator ,
[out,retval] double* pdPosition)
HRESULT **put_CollimatorPosition** ([in] long lCollimator ,
[in] double dPosition)

The "CollimatorPosition" property specifies the position of the servo of the specified collimator. When the position is set the specified position is written to the calibration file.

lCollimator	[in]	The zero based index of the collimator. This is the same index as used for the collimator methods of the hardware information interface.
dPosition	[in]	The new collimator servo position.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetCollimatorAdjustedPosition** ([in] long lCollimator ,
[out,retval] double* pdPosition)

The "GetCollimatorAdjustedPosition" property specifies the position of the servo of the specified collimator where the pinholes have been adjusted.

lCollimator	[in]	The zero based index of the collimator. This is the same index as used for the collimator methods of the hardware information interface.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DetectorTriggerLevel** ([in] long lDetector ,
[out,retval] double* pdLevel)
HRESULT **put_DetectorTriggerLevel** ([in] long lDetector ,
[in] double dLevel)

The "DetectorTriggerLevel" property specifies the value of the DCA for the adjustment of the GaAsP deteector trigger level. The "GetDetectorMinimumTiggerLevel" and "GetDetectorMaximumTiggerLevel" methods of the hardware informations interface can be used to determine the possible adjustment range.

lDetector	[in]	The zero based index of the detector. This is the same index as used for the detector methods of the hardware information interface.
pdLevel	[out]	A pointer to a variable which will receive the current DAC value.
dLevel	[in]	The new DAC value.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsController

The "IAimFcsController" inteface is the root interface of the component. All other interfaces are acessible form the root interface. The interface itself contains the methods to start acquisitions, methods to obtain informations about the acquisition progressand methods to obtain pointers to other interfaces.

Methods

HRESULT **Initialize** ([in] IUnknown* pHarwareControl)

The "Initialize" method creates the object tree of the FCS controller. The "Initialize" method must be the first method called for a new FCS controller object.

pHarwareControl	[in]	A pointer to the main interface of the hardware control program.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **IsAcquisitionRunning** ([out, retval] VARIANT_BOOL* pbAcquisitionRunning)

The "IsAcquisitionRunning" method can be used to determine if a caquisition of FCS data is currently running. This includes the measurement accroding to the acquisition parameters, the acquisition for pinhole adjustement and the scan of countrate images. The online garb of online display values is not considered.

pbAcquisitionRunning	[out]	A pointer to a variable which will receive VARIANT_TRUE if an acquisition is running and VARIANT_FALSE if not.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **StopAcquisition** ( )

The "StopAcquisition" method aborts a running data acquisition. This includes the measurement accroding to the acquisition parameters, the acquisition for pinhole adjustement and the scan of countrate images. The online garb of online display values is not considered. The method will not wait for the acquisition to be stopped. The "IsAcquisitionRunning" method can be used to detemine if an acquisition is stopped.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **StopAcquisitionAndWait** ( )

The "StopAcquisition" method aborts a running data acquisition. This includes the measurement accroding to the acquisition parameters, the acquisition for pinhole adjustement and the scan of countrate images. The online garb of online display values is not considered. The method will wait for the acquisition to be stopped.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **AcquisitionParameters**
([out, retval] IAimFcsAcquisitionParameters** ppAcquisitionParameters)

The "AcquisitionParameters" method yields the interface for the acquisition protocol properties of the acquisition parameters.

ppAcquisitionParameters	[out]	A pointer to a variable which will receive the interface for the acquisition protocol properties.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **BeamPathParameters**
([out, retval] IAimFcsBeamPathParameters** ppBeamPathParameters)

The "BeamPathParameters" method yields the interface for the beam path properties which are used during acquisition.

ppBeamPathParameters	[out]	A pointer to a variable which will receive the interface for the beam path properties.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **SamplePositionParameters**
([out, retval] IAimFcsSamplePositionParameters** ppSamplePositionParameters)

The "SamplePositionParameters" method yields the interface for the acquisition parameters for LSM sample positions and sample carrier.

ppSamplePositionParameters	[out]	A pointer to a variable which will receive the interface for the sample position acquisition parameters.

Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **ProcessingParameters**
([out, retval] IAimFcsProcessingParameters** ppProcessingParameters)

The "ProcessingParameters" method yields the interface for the online processing acquisition parameters.

ppProcessingParameters	[out]	A pointer to a variable which will receive the interface for the online processing acquisition parameters.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **HardwareInformation**
([out, retval] IAimFcsHardwareInformation** ppHardwareInformation)

The "HardwareInformation" method yields the interface with informations about the acquisition hardware.

ppHardwareInformation	[out]	A pointer to a variable which will receive the interface for the informations about the acquisition hardware.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **HardwareControl**
([out, retval] IAimFcsHardwareControl** ppHardwareControl)

The "HardwareControl" method yields the interface for control of hardware components independed of the acquisition process.

ppHardwareControl	[out]	A pointer to a variable which will receive the interface for simple hardware control tasks.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Options** ([out, retval] IAimFcsControllerOptions** ppOptions)

The "Options" method yields the interface for global settings which influence the behaviour of the acquisition routines.

ppOptions	[out]	A pointer to a variable which will receive the interface for global FCS

controller settings.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Calibration** ([out, retval] IAimFcsCalibration** ppCalibration)

The "Calibration" method yields the interface for calibration routines and calibration data.

ppCalibration [out] A pointer to a variable which will receive the interface for calibration routines and calibration data.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Progress** ([out, retval] IDispatch** ppProgress)

The "Progress" method is provided for clients who are not able of calling "QueryInterface" to access other interfaces of the FCS controller and yields the interface for the progress information object for the acquisition operations.

ppProgress [out] A pointer to a variable which will receive the interface of the progress information object.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **StartMeasurement** ([in] IUnknown* pIDestination)

The "StartMeasurement" method starts the data acquisition according to the acquisition parameters for all specified kinetic, position and repeat indexes.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **StartMeasurementOnePosition** ([in] long lPosition ,
[in] IUnknown* pIDestination)

The "StartMeasurementOnePosition" method starts the data acquisition according to the acquisition parameters for all kinetic and repeat indexes but for the specified single position only.

lPosition [in] The zero based index of the measurement position. A negative value indicates that the current focus position is used.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error

information.

HRESULT **GetMeasurementProgressCoordinates** ([out] long* plKinetic ,
[out] long* plPosition ,
[out] long* plRepeat)

The "GetMeasureProgressCoordinates" method determines the kinetics, position nad repeat coordinates of the currently running measurement. If no measurement is running the coordinates where the previous measurement has stopped are returned.

plKinetic	[out]	A pointer to a variable which will receive the current zero based kinetics index.
plPosition	[out]	A pointer to a variable which will receive the current zero based position index.
plRepeat	[out]	A pointer to a variable which will receive the current zero based repeat index.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Backup** (IUnknown* pDestination)

The "Backup" method transfers the acquisition parameters form the parameter set of the fcs controller to an result data object. The amout of transfered data depends on the interfaces which are supported by the destination object.

pDestination	[in]	A pointer to one interface of the result data object where the parameters should be stored. The "Backup" method will look for the following interfaces: IAimFcsDataBeamPath - only the beam path properties are copied IAimFcsDataSamplePositions - only the sample position properties are copied IAimFcsDataAcquisitionParameters - Genaral acquisition and processing parameters as well as the beam path and sample position parameters are copied.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Restore** ([in] IUnknown* pSource ,
[out] BSTR* pMessages)

The "Restore" method transfers the acquisition parameters form a result data object to the parameter set of the fcs controller. The amout of transfered data depends on the interfaces which are supported by the source object.

pSource	[in]	A pointer to one interface of the result data object with the parameters to restore. The "Restore" method will look for the following interfaces: IAimFcsDataBeamPath - only the beam path properties are copied IAimFcsDataSamplePositions - only the sample position properties are copied IAimFcsDataAcquisitionParameters - Genaral acquisition and processing parameters as well as the beam path and sample position

		parameters are copied.
pMessages	[out]	A pointer to a variable which will receive a pointer to the character string with the messages for refered hardware, which is not present.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **SetOnlineDisplayMode** ([in] long lChannel ,
[in] enumOnlineDisplayMode eMode)

The "SetOnlineDisplayMode" method selects the type of data which are calculated by the FCS controller for online display of count rates, correlations or counts per molecule. One of the modes switches the calculation off. The data acquisition is started when the calculation of online display values is switched on and no acquisition is currently running. The data acquisition is stopped when the calculation of online display values is switched off for all channels and no other process (like a measurement) requires data from the detectors.

lChannel	[in]	The zero based index of the channel where the mode should be changed. The index is the same as the index in the hardware information object. The hardware information object can be used to determine the type of the channel.
eMode	[in]	The new mode for calculation of online display values: eOnlineDisplayModeOff - no calculation eOnlineDisplayModeCountRate - calculation of count rate eOnlineDisplayModeCorrelation - calculation of correlation eOnlineDisplayModeCountsPerMolecule - calculation of counts per molecule
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **SetOnlineDisplayAverageTime** ([in] double dTime)

The "SetOnlineDisplayAverageTime" method specifies the time for which data for online display are averaged.

dTime	[in]	The new average time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **StartScanCountRateImage** ([in] double dStartX ,
[in] double dStartY ,
[in] double dStartZ ,
[in] double dEndX ,
[in] double dEndY ,
[in] double dEndZ ,
[in] long lPositionsX ,
[in] long lPositionsY ,
[in] long lPositionsZ)

The "Start" method starts the data acquisition for a count rate image.dStartX dStartY

dStartZ	[in]	The stage and focus positions for the acquisition of the first count rate result.
dEndZ	[in]	The stage and focus positions for the acquisition of the last count rate result.
lPositionsZ	[in]	The number of pixles to acquire in the corresponding stage and focus directions.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetCountRateImageResultArraySize** ([out, retval] long* plSize)

The "GetCountRateImageResultArraySize" method can be used to determine the number of count rate value acquired so far for a count rate image acquisition.

plSize	[out]	A pointer to a variable which will receive the number of acquired count rate values.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetCountRateImageData** ([in] long lChannel ,
[in] long lStart ,
[in] long lSize ,
[out,size_is(Size)] double* pdData)

The "GetCountRateImageData" method copies a region of the resulting of a count rate image acquisition the provided buffer.

lChannel	[in]	The zero based index of the detector channel of the requested data.
lStart	[in]	The zero based index of the first count rate value to copy. The index counting is done in x-direction first. Then follow the values for the next lines (y-direction). After the first plane follow the other planes (z-direction).
lSize	[in]	The number of countrate values to copy.
pdData	[out]	A pointer to the array which will receive the count rate values.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **put_TraceFlags** ([in] long lFlags)
HRESULT **get_TraceFlags** ([out, retval] long* plFlags)

The "TraceFlags" property specifies which actions are recorded to the trace buffer of the FCS controller. The flags are stored to the system registry and re-used in the next run of the program.

lFlags	[in]	A combination of flags, which indicate the type of information that is recorded: eFcsHardwareTraceError - Unexpected error in the FCS controller or the hardware control program reported an error. eFcsHardwareTraceControl - Conntrol instructions sent to the hardware control program. eFcsHardwareTraceInfo - Calls to the hardware control program that are made to get informations about the hardware. eFcsHardwareTraceCall - Calls to the

		hardware control program that are made to obtain state informations. eFcsHardwareTraceBusy - Calls to the hardware control program that are made to obtain the informations, which of the hardware objects are busy. eFcsHardwareTraceReceiver - Calls to the hardware for the handling of the received data.
plFlags	[out]	A pointer to a variable which will receive the currently used flags.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

STDMETHODIMP **TraceGet** ([out,retval] BSTR* pTraceString)

The "TraceGet" method provides the a string with the recorded informations since the previous call "TraceGet" The type of informations to be traced can be selected with the "TraceFlags" property.

pTraceString	[out]	A pointer to a variable which will receive a pointer to the character string with the recorded informations.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **SetTimer** ([in] long lEventNumber ,
[in] double dPeriod)

The "SetTimer" method can be used to enable the generation of timer events from the FCS controller to upper level components. If the timer is enabled the "FireEvent" method of the event interface is called periodically. The method is intended for use im VBA macros.

lEventNumber	[in]	The "lEventNumber" parameter for the event procedure. The value "0" indicates that no events should be generated.
dPeriod	[in]	The timer period in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsOptions

The "IAimFcsOptions" interface is the interface for user specific options which influence the behaviour of the FCS controller independend of the acquisition parameters. This includes the selection of the device for lateral focus positioning, the properties for movement during measure, the uaseage of laser shutter filter wheels and the properties for the handling of AOM-driver blanking signals. The options are stpred user specific in the system registry.

Methods

HRESULT **get_MoveDuringMeasurement**
([out, retval] VARIANT_BOOL* pbMoveDuringMeasurement)
HRESULT **put_MoveDuringMeasurement**
([in] VARIANT_BOOL bMoveDuringMeasurement)

The "MoveDuringMeasurement" property specifies if the lateral focus position is moved during the data acquisition with the currently selected stage.

pbMoveDuringMeasurement	[out]	A pointer to a variable which will receive the current state of the flag.
bMoveDuringMeasurement	[in]	A value different from zero indicates that the stage is moved during the data acquisition. The value zero indicates that the stage is not moved.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MoveDuringMeasurementSpeedIndex**
([out, retval] long* plSpeedIndex)
HRESULT **put_MoveDuringMeasurementSpeedIndex**
([in] long lSpeedIndex)

The "MoveDuringMeasurementSpeedIndex" property specifies the stage speed index which is used for movement of the stage during data acquisition.

plSpeedIndex	[out]	A pointer to a variable which will receive the currently used stage speed index for movement during data acquisition.
lSpeedIndex	[in]	The zero based index of the speed of the currently selected stage. The corresponding velocity can be determined via the hardware information object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error

information.

HRESULT **get_MoveDuringMeasurementDistanceX** ([out, retval] double* pdDistance)
HRESULT **put_MoveDuringMeasurementDistanceX** ([in] double dDistance)

The "MoveDuringMeasurementDistanceX" property specifies the end to end distance of the stage movement during data acquisition in x-direction.

pdDistance	[out]	A pointer to a variable which will receive the currently used distance.
dDistance	[in]	The distance for stage movement during acquisition in x-direction in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MoveDuringMeasurementDistanceY** ([out, retval] double* pdDistance)
HRESULT **put_MoveDuringMeasurementDistanceY** ([in] double dDistance)

The "MoveDuringMeasurementDistanceY" property specifies the end to end distance of the stage movement during data acquisition in y-direction.

pdDistance	[out]	A pointer to a variable which will receive the currently used distance.
dDistance	[in]	The distance for stage movement during acquisition in y-direction in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_StageIndex** ([out, retval] long* plStageIndex)
HRESULT **put_StageIndex** ([in] long lStageIndex)

The "StageIndex" property specifies the zero based index of the lateral focus positioning device in the list of the hardware information object wich is currently used for lateral focus movement. The special value -1 indicates that none of the lateral focus positioning devices is used.

plStageIndex	[out]	A pointer to a variable which will receive the zero based index of the currently used lateral focus positioning device.
lStageIndex	[in]	The zero based index of the lateral focus positioning device to use.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_EnableLaserShutterFilterWheel**
([out, retval] VARIANT_BOOL* pbEnable)
HRESULT **put_EnableLaserShutterFilterWheel**
([in] VARIANT_BOOL bEnable)

The "EnableLaserShutterFilterWheel" property specifies if the acquistion controller can use filter sets in the laser module to block light form laser laser lines that are not used in the acquisition parameters.

pbEnable	[out]	A pointer to a variable which will receive the current state of the setting.
bEnable	[in]	A value different form zero indicates that the filter sets should be used to block laser lines. The value zero indicates that the filtersets should be set to the position of maximal transmission for all laser lines.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticChangeWavelengths**
([out, retval] VARIANT_BOOL* pbChange)
HRESULT **put_AutomaticChangeWavelengths**
([in] BOOL bChange)

The "AutomaticChangeWavelengths" property specifies if the acquistion controller should automatically control the laser and attenuator driver wavelength when the acquisition parameters contain settings for wavelengths which are currently not selected.

pbChange	[out]	A pointer to a variable which will receive the current state of the setting.
bChange	[in]	A value different form zero indicates that hardware is controlled when the acquisition parameters contain settings for wavelengths which are currently not selected. The value zero indicates that the hardware is not controlled.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AomPower** ([out, retval] double* pdAomPower)
HRESULT **put_AomPower** ([in] double dAomPower)

The "AomPower" property specifies the power of the attenuator driver blanking signal for the case that the "AutomaticChangeWavelengths" property is set to FALSE.

pdAomPower	[out]	A pointer to a variable which will receive the currently used power.
dAomPower	[in]	the power of the attenuator driver blanking signal in the range 0..1.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SpotIlluminationPause** ([out, retval] double* pdPause)
HRESULT **put_SpotIlluminationPause** ([in] double dPause)

The "SpotIlluminationPause" property specifies the pause between scanner movement and illumination in fast sequential acqusition mode for different scanner positions.

pdPause	[out]	A pointer to a variable which will receive the currently used pause between between scanner movement and ilumination in seconds.
dPause	[in]	The pause between scanner movement and ilumination in seconds.

Returns "S_OK" if successful or an error code. In the latter case the "LastError"

method for the component can be used to obtain detailed error information.

HRESULT **get_SequentialIlluminationPause** ([out, retval] double* pdPause)
HRESULT **put_SequentialIlluminationPause** ([in] double dPause)

The "SequentialIlluminationPause" property specifies the pause between the illumination phases in fast sequential acqusition mode.

pdPause	[out]	A pointer to a variable which will receive the pause between the illumination phases in seconds.
dPause	[in]	The pause between the illumination phases in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_StartMarkerDelay** ([out, retval] double* pdStartMarkerDelay)
HRESULT **put_StartMarkerDelay** ([in] double dStartMarkerDelay)

The "StartMarkerDelay" property specifies the delay between illumination switching to "on" state and marker generation in fast sequential acqusition for different positions or illuminations.

pdStartMarkerDelay	[out]	A pointer to a variable which will receive the currently used delay between illumination switching and marker generation in seconds.
dStartMarkerDelay	[in]	The delay between illumination switching and marker generation in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_EndMarkerDelay** ([out, retval] double* pdEndMarkerDelay)
HRESULT **put_EndMarkerDelay** ([in] double dEndMarkerDelay)

The "EndMarkerDelay" property specifies the delay between illumination switching to "off" state and marker generation in fast sequential acqusition for different positions or illuminations.

pdEndMarkerDelay	[out]	A pointer to a variable which will receive the currently used delay between illumination switching and marker generation in seconds.
dEndMarkerDelay	[in]	The delay between illumination switching and marker generation in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutoSave** ([out, retval] VARIANT_BOOL* pbAutoSave)
HRESULT **put_AutoSave** ([in] VARIANT_BOOL bAutoSave)

The "AutoSave" property specifies if the acquiusition results are written drirectly to a file during acquisition. The name of the file is generated form the the path name in the "AutoSavePath" property.

pbAutoSave	[out]	A pointer to a variable which will receive the current state of the flag.
bAutoSave	[in]	The value zero indicates that the acquisition results are not written to a file. A different value indicates that the results are written to a file.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **put_AutoSavePath** ([in] BSTR pAutoSavePath)
HRESULT **get_AutoSavePath** ([out, retval] BSTR* ppAutoSavePath)

The "AutoSavePath" method specifies the path where the result data are written during data acquisition. Numbers are appended to the specified file name part (without extension) to create a unique file name.

pAutoSavePath	[in]	The new path for the destination file.
ppAutoSavePath	[in]	A reference to a variable which will receive a pointer to the character string with the currently used destination path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SaveRawData** ([out, retval] VARIANT_BOOL* pSaveRawData)
HRESULT **put_SaveRawData** ([in] VARIANT_BOOL bSaveRawData)

The "SaveRawData" property specifies if the raw data are written to a file during acquisition. The name of the file is generated form the the path name in the "RawDataPath" property.

bSaveRawData	[in]	The value zero indicates that the raw data are not written to a file. A different value indicates that the raw data are written to a file.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **put_RawDataPath** ([in] BSTR pRawDataPath)
HRESULT **get_RawDataPath** ([out, retval] BSTR* ppRawDataPath)

The "RawDataPath" method specifies the path where the result data are written during data acquisition. Numbers are appended to the specified file name part (without extension) to create a unique file name.

pRawDataPath	[in]	The new path for the destination file.
ppRawDataPath	[in]	A reference to a variable which will receive a pointer to the character string with the currently used destination path.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsHardwareControl

The "IAimFcsHardwareControl" interface is the interface for control tasks form the user interface independed of the measurement and adjutment exectives of the FCS controller. The interface can be used to move stage and focus and to change the beam path mode of the microscope (LSM, FCS, VIS etc.).

Methods

HRESULT **MoveToMeasurementPosition** ([in] long lPosition)

The "MoveToMeasurementPosition" method moves the focus to the lateral and axial position which corresponds to the measurement position specified by index.

lPosition	[in]	The zero based index of the position.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStagePosition** ([out] double* pdPositionX ,
[out] double* pdPositionY)

The "GetCurrentStagePosition" can be used to determine the current position of the lateral focus positioning device which is currently selected.pdPositionX

pdPositionY	[out]	The pointers to variables which will receive the current position coordinates in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **MoveStageToPosition** ([in] double dPositionX ,
[in] double dPositionY)

The "MoveStageToPosition" method moves the specified lateral focus positioning device which is currently selected to the specified position.dPositionX

dPositionY	[in]	The position coordinates in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **LockStageDrives** ([in] VARIANT_BOOL bLock)

The "LockStageDrives" method locks the stage drives of the currently selected stage for control by software or unlocks the stage drives for use with the hand wheels.

bLock	[in]	The value 0 unlocks the drives (handwheel). A different value locks the drives (control by software).
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_StageSpeedIndex** ([out, retval] long* plIndex)
HRESULT **put_StageSpeedIndex** ([in] long lIndex)

The "StageSpeedIndex" property specifies the speed of the currently selected lateral focus positioning device by the index in the list of predefined speeds.

plIndex	[out]	A pointer to a variable which will receive the currently selected speed index.
lIndex	[in]	The zero based index of the speed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **IsStageBusy** ([out, retval] VARIANT_BOOL* pbBusy)

The "IsStageBusy" method can be called to find out if the process to control the currently selected lateral focus positioning device is still running.

pbBusy	[out]	A pointer to a variable which will receive VARIANT_TRUE if the lateral focus positioning device is busy and VARIANT_FALSE if not.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetFocusPosition** ([out] double* pdPositionZ)

The "FocusGetPosition" method reads the current position of the axial focus positioning device.

pdPositionZ	[in]	A pointer to a variable that will receive the position in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **MoveFocusToPosition** ([in] double dPositionZ)

The "MoveFocusToPosition" method moves the axial focus positioning device to the specified position. The instruction is sent to the hardware, but the method will not wait for the hardware to

finish the control task. One can call the "IsFocusBusy" method to wait for completition of the control task.

dPositionZ	[in]	The destination position in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **MoveFocusToLoadPosition** ( )

The "MoveFocusToWorkPosition" methods moves the axial focus positioning device to the sample load position. The instruction is sent to the hardware, but the method will not wait for the hardware to finish the control task. One can call the "IsFocusBusy" method to wait for completition of the control task.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **MoveFocusToWorkPosition** ( )

The "FocusMoveToWorkPosition" methods moves the axial focus positioning device to the work position. The instruction is sent to the hardware, but the method will not wait for the hardware to finish the control task. One can call the "IsFocusBusy" method to wait for completition of the control task.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **IsFocusBusy** (BOOL& bBusy)

The "IsFocusBusy" method can be called to find out if the process to control the axial focus positioning device is currently running.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ObjectiveRevolverPosition** ([out, retval] long* plPosition)
HRESULT **put_ObjectiveRevolverPosition** ([in] long lPosition)

The "ObjectiveRevolverPosition" property specifies the index of the position of the current position in the objective revolver.

plPosition	[out]	A pointer to a variable that will receive the zero based index of the objective revolver position. The value -1 is written to the variable if there is no objective revolver or the objective revolver is in an intermediate position.
lPosition	[in]	The new zero based index of the objective revolver position.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MicroscopeMode** ([out, retval] enumFcsMicroscopeMode* peMode)
HRESULT **put_MicroscopeMode** ([in] enumFcsMicroscopeMode eMode)

The "MicroscopeMode" property specifies the microscope system mode (LSM, VIS, TV, FCS or FCS camera) which is controlled with sliders and filtersets in the microscope stand and FCS head. When the property is set the corresponding sliders and filtersets are controlled.

peMode	[out]	A pointer to a variable which will receive the current microscope mode:
eMode	[in]	The new microscope mode: eFcsMicroscopeModeLSM - LSM port eFcsMicroscopeModeFCS - FCS port eFcsMicroscopeModeVIS - conventional microscpy eFcsMicroscopeModeTV - TV camera port eFcsMicroscopeModeCamera - FCS camera for sample adjustement eFcsMicroscopeModeUnknown - intermediate state of other (unknown) mode.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsHardwareInformation

The "IAimFcsHardwareInformation" interface is the interface where upper level components can obtain their informations about the hardware which is used by the FCS controller. All informations are independendend of the current acquisition parameters.

Methods

HRESULT **GetNumberChannels** ([out,retval] long* plCount)

The "GetNumberChannels" method can be used to obtain the number of FCS detection channels in the system.

plCount	[out]	A pointer to a variable which will receive the number of channels.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetChannelType** ([in] long lChannel ,
[out,retval] enumFcsChannelIdentifier* peChannelType)

The "GetChannelType" can be used to determine if a channel is an auto or cross correlation channel.

lChannel	[in]	The zero based index of the channel.
peChannelType	[out]	A pointer to a variable which will receive the type of the channel: eFcsChannelAutoCorrelation - auto correlation eFcsChannelCrossCorrelation - cross correlation eFcsChannelInvalid - There is no such channel.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetChannelIdentifier**
([in] long lChannel ,
[out, retval] enumFcsChannelIdentifier* peChannelIdentifier)

The "GetChannelIdentifier" method provides a unique identfier for a data channel specified by list index.

lChannel	[in]	The zero based index of the channel.

peChannelIdentifier	[out]	A pointer to a variable which will receive the identifier of the channel: eFcsChannelAutoCorrelation1 - auto correlation with data of frist detector. eFcsChannelAutoCorrelation2 - auto correlation with data of second detector. eFcsChannelCrossCorrelation1Versus2 - cross correlation with data of first detector versus second detector. eFcsChannelCrossCorrelation2Versus1 - cross correlation with data of second detector versus first detector. eFcsChannelInvalid - There is no such channel.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **ChannelIndexFromIdentifier**

([in] enumFcsChannelIdentifier eChannelIdentifier ,
[out, retval] long* plChannelIndex)

The "ChannelIndexFromIdentifier" method can be used to determine the list index of a channel from the channel identifier.

eChannelIdentifier	[in]	The Identifier of the channel. See "GetChannelIdentifier" for possible values.
plChannelIndex	[out]	A pointer to a variable which will receive the zero based list index in the channels array or -1 if there is no such channel.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberDetectors** ([out, retval] long* plCount)

The "GetNumberDetectors" method can be used to obtain the number of FCS detectors in the system.

plCount	[out]	A pointer to a variable which will receive the number of detectors.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

STDMETHOD **GetDetectorType**

([in] long lDetector ,
[out, retval] enumFcsDetectorIdentifier* peDetectorType)

The "GetDetectorType" returns a value that identifies the type of a FCS detector.

lDetector	[in]	The zero based index of the detector.
peDetectorType	[out]	A pointer to a variable which will receive the type of the detector: eFcsDetectorApd - avalache photo diode eFcsDetectorGaAsp - GaAsP detector. eFcsDetectorInvalid - There is no such detector.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetDetectorIdentifier**

([in] long lDetector ,
[out, retval] enumFcsDetectorIdentifier* peDetectorIdentifier)

The "GetDetectorIdentifier" method provides a unique identfier for a FCS detector.

lDetector	[in]	The zero based index of the detector.
peDetectorIdentifier	[out]	A pointer to a variable which will receive the identifier of the detector: eFcsDetectorApd1 - The first avalache photo diode. eFcsDetectorApd2 - The second avalache photo diode. eFcsDetectorGaAsp1 - The first GaAsP detector. eFcsDetectorGaAsp2 - The second GaAsP detector. eFcsDetectorInvalid - There is no such detector.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **DetectorIndexFromIdentifier**

([in] enumFcsDetectorIdentifier eDetectorIdentifier ,
[out, retval] long* plDetectorIndex)

The "DetectorIndexFromIdentifier" method can be used to determine the list index of a detector from the detector identifier.

eDetectorIdentifier	[in]	The Identifier of the detector. See "GetDetectorIdentifier" for possible values.
plDetectorIndex	[out]	A pointer to a variable which will receive the zero based list index in the detectors array or -1 if there is no such detector.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetDetectorMinimumVoltage** ([in] long lDetector ,
[out, retval] double* pdMinimumVoltage)

The "GetDetectorMinimumVoltage" method can be used to determine the minimum controllable voltage for the detector specified by index.

lDetector	[in]	The zero based index in the list of detectors.
pdMinimumVoltage	[out]	A pointer to a variable which will receive the minimum controllable voltage. The value 0.0 is written to the variable if there is no such detector or no controllable voltage for the detector.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetDetectorMaximumVoltage** ([in] long lDetector ,
[out, retval] double* pdMaximumVoltage)

The "GetDetectorMaximumVoltage" method can be used to determine the maximum controllable voltage for the detector specified by index.

lDetector	[in]	The zero based index in the list of detectors.
pdMaximumVoltage	[out]	A pointer to a variable which will receive the maximum controllable voltage. The value 0.0 is written to the variable if there is no such detector or no controllable voltage for the detector.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetDetectorMinimumTiggerLevel**
([in] long lDetector ,
[out, retval] double* pdMinimumTiggerLevel)

The "GetDetectorMinimumTriggerLevel" method can be used to determine the minimum controllable value for the DAC which sets the trigger level of an GaAsP detector.

lDetector	[in]	The zero based index in the list of detectors.
pdMinimumTiggerLevel	[out]	A pointer to a variable which will receive the minimum controllable DAC value for the GaAsP detector trigger level. The value 0.0 is written to the variable if there is no such detector or no such DAC for the detector.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetDetectorMaximumTiggerLevel**
([in] long lDetector ,
[out, retval] double* pdMaximumTiggerLevel)

The "GetDetectorMaximumTriggerLevel" method can be used to determine the maximum controllable value for the DAC which sets the trigger level of an GaAsP detector.

lDetector	[in]	The zero based index in the list of detectors.
pdMaximumTiggerLevel	[out]	A pointer to a variable which will receive the maximum controllable DAC value for the GaAsP detector trigger level. The value 0.0 is written to the variable if there is no such detector or no such DAC for the detector.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberPinholes** ([out, retval] long* plCount)

The "GetNumberPinholes" method can be used to obtain the number of pinholes in the system which are relevant for FCS data acquisition.

plCount	[out]	A pointer to a variable which will receive the number of pinholes.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeType** ([in] long lPinhole ,
[out, retval] enumFcsPinholeIdentifier* pePinholeType)

enumFcsPinholeIdentifier GetPinholeType (long lPinhole)The "GetPinholeType" method can be used to determine if a pinhole is exclusively used for one channel or used for multiple channels.

lPinhole	[in]	The zero based index of the pinhole.
pePinholeType	[out]	A pointer to a variable which will receive the type of the pinhole: eFcsDetectorPinhole - a single detector pinhole eFcsDetectorsPinhole - a pinhole for multiple detectors eFcsPinholeInvalid - there is no such pinhole

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeIdentifier**
([in] long lPinhole ,
[out, retval] enumFcsPinholeIdentifier* pePinholeIdentifier)

The "GetChannelIdentifier" method provides a unique identfier for a pinhole specified by list index.

lPinhole	[in]	The zero based index of the pinhole.
pePinholeIdentifier	[out]	A pointer to a variable which will receive the identifier of the pinhole: eFcsDetectorPinhole1 - a pinhole exclusive for detector 1 eFcsDetectorPinhole2 - a pinhole exclusive for detector 2 eFcsDetectorsPinhole1_2 - a pinhole for detector 1 and 2 eFcsPinholeInvalid - there is no such pinhole

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PinholeIndexFromIdentifier**

([in] enumFcsPinholeIdentifier ePinholeIdentifier ,
[out, retval] long* plPinholeIndex)

The "PinholeIndexFromIdentifier" method can be used to determine the list index of a pinhole from the pinhole identifier.

ePinholeIdentifier	[in]	The Identifier of the pinhole. See "GetPinholeIdentifier" for possible values.
plPinholeIndex	[out]	A pointer to a variable which will receive the zero based list index in the pinhole array or -1 if there is no such pinhole.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMinimumDiameter** ([in] long lPinhole ,
[out, retval] double* pdMinValue)

The "GetPinholeMinimumDiameter" method can be used to determine the minimum controllable diameter of a pinhole.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMinValue	[out]	A pointer to a variable which will receive the minimum controllable pinhole diameter in metre. The value 0.0 is written to the variable if there is no such pinhole.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMaximumDiameter** ([in] long lPinhole ,
[out, retval] double* pdMaxValue)

The "GetPinholeMaximumDiameter" method can be used to determine the maximum controllable diameter of a pinhole.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMaxValue	[out]	A pointer to a variable which will receive the maximum controllable pinhole diameter in metre. The value 0.0 is written to the variable if there is no such pinhole.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMinimumPositionX** ([in] long lPinhole ,
[out, retval] double* pdMinValue)

The "GetPinholeMinimumPositionX" method can be used to determine the minimum controllable position of a pinhole x-servo.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMinValue	[out]	A pointer to a variable which will receive the minimum controllable position or 0.0 if there is no such pinhole or if the x-position is not controllable.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMaximumPositionX** ([in] long lPinhole ,
[out, retval] double* pdMaxValue)

The "GetPinholeMaximumPositionX" method can be used to determine the maximum controllable position of a pinhole x-servo.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMaxValue	[out]	A pointer to a variable which will receive the maximum controllable position or 0.0 if there is no such pinhole or if the x-position is not controllable.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMinimumPositionY** ([in] long lPinhole ,
[out, retval] double* pdMinValue)

The "GetPinholeMinimumPositionY" method can be used to determine the minimum controllable position of a pinhole y-servo.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMinValue	[out]	A pointer to a variable which will receive the minimum controllable position or 0.0 if there is no such pinhole or if the y-position is not controllable.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMaximumPositionY** ([in] long lPinhole ,
[out, retval] double* pdMaxValue)

The "GetPinholeMaximumPositionY" method can be used to determine the maximum controllable position of a pinhole y-servo.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMaxValue	[out]	A pointer to a variable which will receive the maximum controllable position or 0.0 if there is no such pinhole or if the y-position is not controllable.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError"

method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMinimumPositionZ** ([in] long lPinhole ,
[out, retval] double* pdMinValue)

The "GetPinholeMinimumPositionZ" method can be used to determine the minimum controllable position of a pinhole z-servo.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMinValue	[out]	A pointer to a variable which will receive the minimum controllable position or 0.0 if there is no such pinhole or if the z-position is not controllable.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetPinholeMaximumPositionZ** ([in] long lPinhole ,
[out, retval] double* pdMaxValue)

The "GetPinholeMaximumPositionZ" method can be used to determine the maximum controllable position of a pinhole z-servo.

lPinhole	[in]	The zero based index in the list of pinholes.
pdMaxValue	[out]	A pointer to a variable which will receive the maximum controllable position or 0.0 if there is no such pinhole or if the z-position is not controllable.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberFilterSets** ([out, retval] long* plCount)

The "GetNumberFilterSets" method can be used to obtain the number of filter sets in the detection beam path of the system which are relevant for FCS data acquisition. This includes beam splitters and emission filters.

plCount	[out]	A pointer to a variable which will receive the number of filter sets.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetFilterSetType** ([in] long lFilterSet ,
[out, retval] enumFcsFilterSetIdentifier* peFilterSetType)

The "GetFilterSetType" method can be used to determine if a filter set is an emission filter for one ro more detectors or a beam splitter.

lFilterSet	[in]	The zero based index of the filter set.
peFilterSetType	[out]	A pointer to a variable which will receive the type of the filter set: eFcsFilterSetEmissionFilterDetector - an emission filter set next to one of the detectors. eFcsFilterSetEmissionFilterDetectors - a filter set where emmision light for multiple detectors transmits. eFcsFilterSetBeamSplitter - a beam splitter eFcsFilterSetInvalid - there is no such filter set
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetFilterSetIdentifier**

([in] long lFilterSet ,
[out, retval] enumFcsFilterSetIdentifier* peFilterSetIdentifier)

The "GetFilterSetIdentifier" method provides a unique identfier for the filter set specified by list index.

lFilterSet	[in]	The zero based index of the filter set.
peFilterSetIdentifier	[out]	A pointer to a variable which will receive the identifier of the filter set: eFcsFilterSetEmissionFilterDetector1 - the emission filter set next to the first detector. eFcsFilterSetEmissionFilterDetector2 - the emission filter set next to the second detector. eFcsFilterSetEmissionFilterDetector1_2 - the filter set where emmision light of the first and second detector transmits. eFcsFilterSetMainBeamSplitter - the main beam splitter eFcsFilterSetBeamSplitterDetectors1_2 - the dichroic beam splitter that seperates the light between FCS detector 1 and 2 eFcsFilterSetLsmBeamSplitterDescanned1 eFcsFilterSetLsmBeamSplitterDescanned2 eFcsFilterSetLsmBeamSplitterDescanned3 - the LSM beam splitter in the FCS detection beam path. eFcsFilterSetInvalid - there is no such filter set
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **FilterSetIndexFromIdentifier**

([in] enumFcsFilterSetIdentifier eFilterSetIdentifier ,
[out, retval] long* plFilterSetIndex)

The "FilterSetIndexFromIdentifier" method can be used to determine the list index of a filter set from the filter set identifier.

eFilterSetIdentifier	[in]	The Identifier of the filter set. See "GetFilterSetIdentifier" for possible values.
plFilterSetIndex	[out]	A pointer to a variable which will receive the zero based list index in the filter set array or -1 if there is no such filter set.
Returns		"S_OK" if successful or an error code. In the latter case the

"LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberFilterSetFilters** ([in] long lFilterSet ,
[out, retval] long* plCount)

The "GetNumberFilterSetFilters" method can be used to determine the number of filters in the filter set specified by list index.

lFilterSet	[in]	The zero based index of the filter set.
plCount	[out]	A pointer to a variable which will receive the number of filters in the filter set. The value 0 is wirtten to the variable if there is no such filter set.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetFilterSetFilterName** ([in] long lFilterSet ,
[in] long lIndex ,
[out, retval] BSTR* pName)

The "GetFilterSetFilterName" method can be used to determine the name of the specified filter in the specified filter set.

lFilterSet	[in]	The zero based index of the filter set.
lIndex	[in]	The zero based position of the filter in the filter set.
pName	[out]	A pointer to a variable which will receive a pointer to the character string with the name of the filter. If there is no such filter an empty string is written.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetFilterSetFilterWavelength** ([in] long lFilterSet ,
[in] long lIndex ,
[out, retval] double* pdWavelength)

The "GetFilterSetFilterWavelength" method can be used to obtain the name of the ceneter wavelength of the specified filter in the specified filter set. The center wavelength returned is intended for display of colors in the user interface.

lFilterSet	[in]	The zero based index of the filter set.
lIndex	[in]	The zero based position of the filter in the filter set.
pdWavelength	[out]	A pointer to a variable which will receive the center wavelength of the filter in metre or 0 if there is no such filter.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberLasers** ([out, retval] long* plCount)

The "GetNumberLasers" method can be used to determine the number of lasers in the FCS beampath of the system.

plCount	[out]	A pointer to a variable which will receive the number of lasers.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetLaserName** ([in] long lLaser ,
[out, retval] BSTR* pName)

The "GetLaserName" method can be used to obtain the name hardware control object for the specified laser.

lLaser	[in]	The zero based index of the laser.
pName	[out]	A pointer to a variable which will receive a pointer to the character string with the name of the laser.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberLaserWavelengths** ([in] long lLaser ,
[out, retval] long* plNumberWavelengths)

The "GetNumberLaserWavelengths" method can be used to determine the number of wavelengths a laser emmits.

lLaser	[in]	The zero based index of the laser.
plNumberWavelengths	[out]	A pointer to a variable which will receive the number of wavelengths.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetLaserWavelength** ([in] long lLaser ,
[in] long lIndex ,
[out, retval] double* pdWavelength)

The "GetLaserWavelength" method can be used to obtain the wavelengths the laser specified by index. Note that the wavelength of a laser can be changeable.

lLaser	[in]	The zero based index of the laser.
lIndex	[in]	The zero based index of the laser line.
pdWavelength	[out]	A pointer to a variable which will receive the wavelength in metre or "0" if there is no such laser or laser line.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error

information.

HRESULT **GetMaximumLaserPower** ([in] long lLaser ,
[out, retval] double* pdPower)

The "GetLaserMaximumPowe" method returns the maximum power for the brigthest laser line of the specified laser.

lLaser	[in]	The zero based index of the laser.
pdPower	[out]	A pointer to a variable which will receive the maximum laser power in Watt.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **IsSequentialIlluminationSupported**
([out, retval] VARIANT_BOOL* pbSupported)

The "IsSequentialIlluminationSupported" method can be used to determine if the actual device supports the switching of atteunator to reduce blead through.

pbSupported	[out]	A pointer to a variable which will receive VARIANT_TRUE if the device supports blead through reduction and VARIANT_FALSE if not.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberCollimators** ([out, retval] long* plCount)

The "GetNumberCollimators" method can be used to determine the number of collimators in the FCS excitation beam path of the system.

plCount	[out]	A pointer to a variable which will receive the number of collimators.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetCollimatorMinimumPosition** ([in] long lCollimator ,
[out, retval] double* pdMinimum)

The "GetCollimatorMinimumPosition" method can be used to determine the minimum controllable position of a collimator servo.

lCollimator	[in]	The zero based index in the list of collimators.
pdMinimum	[out]	A pointer to a variable which will receive the minimum controllable position or 0.0 if there is no such collimator.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error

information.

HRESULT **GetCollimatorMaximumPosition** ([in] long lCollimator ,
[out, retval] double* pdMaximum)

The "GetCollimatorMaximumPosition" method can be used to determine the maximum controllable position of a collimator servo.

lCollimator	[in]	The zero based index in the list of collimators.
pdMaximum	[out]	A pointer to a variable which will receive the maximum controllable position or 0.0 if there is no such collimator.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetCollimatorMinimumWavelength** ([in] long lCollimator ,
[out, retval] double* pdMinimum)

The "GetCollimatorMinimumWavelength" method can be used to determine the minimum of wavelength of the lasers for which the laser beam transmits the specified collimator.

lCollimator	[in]	The zero based index in the list of collimators.
pdMinimum	[out]	A pointer to a variable which will receive the minimum laser wavelength or 0.0 if there is no such collimator.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetCollimatorMaximumWavelength** ([in] long lCollimator ,
[out, retval] double* pdMaximum)

The "GetCollimatorMaximumWavelength" method can be used to determine the maximum of wavelength of the lasers for which the laser beam transmits the specified collimator.

lCollimator	[in]	The zero based index in the list of collimators.
pdMaximum	[out]	A pointer to a variable which will receive the maximum laser wavelength or 0.0 if there is no such collimator.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberAttenuators** ([out, retval] long* plCount)

The "GetNumberAttenuators" method can be used to obtain the number of attenuators in the FCS excitation beam path of the system.

plCount	[out]	A pointer to a variable which will receive the number of attenuators.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetAttenuatorWavelength** ([in] long lAttenuator ,
[out, retval] double* pdWavelength)

The "GetAttenuatorWavelength" method can be used to obtain the wavelength of the attenuator specified by index. Note that the wavelength is changeable on some devices.

lAttenuator	[in]	The zero based index of the attenuator.
pdWavelength	[out]	A pointer to a variable which will receive the wavelength in metre or 0.0 if there is no such attenuator.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetAttenuatorMinimumWavelength** ([in] long lAttenuator ,
[out, retval] double* pdMinimum)

The "GetAttenuatorMinimumWavelength" method can be used to obtain the minimum wavelength that is controllable for the attenuator.

lAttenuator	[in]	The zero based index of the attenuator.
pdMinimum	[out]	A pointer to a variable which will receive the minimum wavelength in metre or 0.0 if there is no such attenuator.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetAttenuatorMaximumWavelength** ([in] long lAttenuator ,
[out, retval] double* pdMaximum)

The "GetAttenuatorMaximumWavelength" method can be used to obtain the maximum wavelength that is controllable for the attenuator.

lAttenuator	[in]	The zero based index of the attenuator.
pdMaximum	[out]	A pointer to a variable which will receive the maximum wavelength in metre or 0.0 if there is no such attenuator.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetAttenuatorNumberLasers** ([in] long lAttenuator ,
[out, retval] long* plLasers)

The "GetAttenuatorNumberLasers" method returns the number of lasers in the beampath of a particular attenuator.

lAttenuator	[in]	The zero based index of the attenuator for which the information is requested.
plLasers	[out]	A pointer to a variable which will receive the number of laserss.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetAttenuatorLaser** ([in] long lAttenuator ,
[in] long lIndex ,
[out, retval] long* plLaser)

The "GetLaserFromAttenuator" method can be used to obtain the information which attenuator is connected to which laser.

lAttenuator	[in]	The zero based index of the attenuator for which the information is requested.
lIndex	[in]	The zero based index in the laser list for the attenuator.
plLaser	[out]	A pointer to a variable which will receive the zero based index of the laser in the same beam path like the attenuator channel.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberStages** ([out, retval] long* plCount)

The "GetNumberStages" method can be used to obtain the number of lateral focus positioning devices.

plCount	[out]	A pointer to a variable which will receive the number of lateral focus positioning devices.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

STDMETHODIMP **GetStageIdentifier**
([in] long lStage ,
[out, retval] enumStageIdentifier* peStageIdentifier)

The "GetStageIdentifier" method provides a value that identifies the type of the of lateral focus positioning device.

lStage	[in]	The zero based index of the lateral focus positioning device.
peStageIdentifier	[out]	A pointer to a variable which will receive the type of the lateral focus positioning device: eFcsStageMotorizedXY - motorized scanning stage eFcsStagePiezoXY - piezo stage eFcsStageScannerXY - galvo scanner in the scan head

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

STDMETHODIMP **StageIndexFromIdentifier** ([in] enumStageIdentifier eStageIdentifier ,
[out, retval] long* plStageIndex)

The "StageIndexFromIdentifier" method can be used to determine the list index of a lateral focus positioning device from the identifier.

eStageIdentifier	[in]	The Identifier of the lateral focus positioning device. See "GetStageIdentifier" for possible values.
plStageIndex	[out]	A pointer to a variable which will receive the zero based list index in the array of lateral focus positioning devicees or -1 if there is no such device.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStageResolution** (long lStage ,
double* pdResolution)

The "GetStageResolution" method can be used to determine the minimum step distance of the specified lateral focus positioning device. An attempt to move the stage a smaller distance will not cause any reaction of the hardware.

lStage	[in]	The zero based index of the lateral focus positioning device in list of the hardware information object.
pdResolution	[out]	A pointer to a variable which will receive the minimum step distance in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStageMinimumPositionX** ([in] long lStage ,
[out, retval] double* pdMinimum)

The "GetStageMinimumPositionX" method can be used to determine the minimum controllable x-position of a lateral focus positioning device.

lStage	[in]	The zero based index of the lateral focus positioning device.
pdMinimum	[out]	A pointer to a variable which will receive the minimum controllable position or 0.0 if there is no such device.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStageMaximumPositionX** ([in] long lStage ,
[out, retval] double* pdMaximum)

The "GetStageMaximumPositionX" method can be used to determine the maximum controllable x-position of a lateral focus positioning device.

lStage	[in]	The zero based index of the lateral focus positioning device.
pdMaximum	[out]	A pointer to a variable which will receive the maximum controllable position or 0.0 if there is no such device.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStageMinimumPositionY** ([in] long lStage ,
[out, retval] double* pdMinimum)

The "GetStageMinimumPositionY" method can be used to determine the minimum controllable y-position of a lateral focus positioning device.

lStage	[in]	The zero based index of the lateral focus positioning device.
pdMinimum	[out]	A pointer to a variable which will receive the minimum controllable position or 0.0 if there is no such device.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStageMaximumPositionY** ([in] long lStage ,
[out, retval] double* pdMaximum)

The "GetStageMaximumPositionY" method can be used to determine the maximum controllable y-position of a lateral focus positioning device.

lStage	[in]	The zero based index of the lateral focus positioning device.
pdMaximum	[out]	A pointer to a variable which will receive the maximum controllable position or 0.0 if there is no such device.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStageNumberSpeeds** ([in] long lStage ,
[out, retval] long* plCount)

The "GetStageNumberSpeeds" method can be used to obtain the number selectable positioning speeds of a lateral focus positioning device.

lStage	[in]	The zero based index of the lateral focus positioning device.
plCount	[out]	A pointer to a variable which will receive the number of selectable positioning speeds. The value zero is returned if the speed is not adjustable.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetStageSpeed** ([in] long lStage ,
[in] long lIndex ,
[out, retval] double* pdSpeed)

The "GetStageSpeed" method can be used to obtain the positioning velocities of the lateral focus positioning device.

lStage	[in]	The zero based index of the lateral focus positioning device.
lIndex	[in]	The zero based index in the list of selectable positioniong speeds.
pdSpeed	[out]	A pointer to a variable which will receive the positioning speed in metre per second.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **IsLaserShutterFilterWheelPresent**
([out,retval] VARIANT_BOOL* pbPresent)

The "IsLaserShutterFilterWheelPresent" method can be used if there is at least one laser shutter filter wheel in the FCS acquisition beam path. The "EnableLaserShutterFilterWheel" option in the options interface has no meaning when no such filterwheel is present.

pbPresent	[out]	A pointer to a variable which will receive VARIANT_TRUE if a shutter filter wheel is present and VARIANT_FALSE if not.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetMicroscopeModeForFcs**
([out,retval] enumFcsMicroscopeMode* peMode)

The "GetMicroscopeModeForFcs" method can be called to determine the microscope mode which has to be selected for a FCS acquisition.

peMode	[out]	A pointer to a variable which will receive the identifier of the microscope mode: eFcsMicroscopeModeLSM - LSM port eFcsMicroscopeModeFCS - FCS port The other constants of "enumFcsMicroscopeMode" are currently not used.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberObjectives** ([out, retval] long* plCount)

The "GetObjectiveRevolverName" method can be used to determine the number of objectives in the objective revolver which can be used for acquisition.

plCount	[out]	A pointer to a variable which will receive the number of objectives.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetObjectiveName** ([in] long lIndex ,
[out, retval] BSTR* pName)

The "GetObjectiveName" method can be used to determine the name of a particular objective in the objective revolver.

lIndex	[in]	The zero based index of the objective in the revolver.
pName	[out]	A pointer to a variable which will receive the string string with the name of the objective.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetObjectiveWorkingDistance** ([in] long lIndex ,
[out, retval] double* pdWorkingDistance)

The "GetObjectiveName" method can be used to determine the distance of the objectives from the focus.

lIndex	[in]	The zero based index of the objective in the revolver.
pdWorkingDistance	[out]	A pointer to a variable which will receive the distance of the objective from the focus.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Dump** ([out,retval] BSTR* pState)

The "Dump" method generates a string with the informations about the hardware components that are controlled by the FCS controller.

pState	[in]	A pointer to a variable which will receive the string with the hardware informations.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsProcessingParameters

The "IAimFcsProcessingParameters" interface is one of four interfaces for acquisition parameters. Via this interface the parameters for correlator and fit are accessible.

Methods

HRESULT **FitParameters** ([in] long lChannel ,
[out, retval] IUnknown** ppFitParameters)

The "FitParameters" method yields the interface pointer for the parameter object for the online fit for the specified acquisition channel.

lChannel	[in]	The zero based index of the detection channel.
ppFitParameters	[out]	A pointer to a variable which will receive the interface pointer for the fit parameters object. The fit parameters object is of type "AimFcsDataFitParameters".

HRESULT **get_BaseLineCorrection**
([out, retval] VARIANT_BOOL* pbBaseLineCorrection)
HRESULT **put_BaseLineCorrection**
([in] VARIANT_BOOL bBaseLineCorrection)

The "BaseLineCorrection" property specifies if the count rate inetensity drop due to bleaching has been corrected during acquisition.

pbBaseLineCorrection	[out]	A pointer to a variable which will receive the current state of the flag.
bBaseLineCorrection	[in]	The value 0 indicates that the count rate data have not been corrected. A different value indicates that the count rate data have been corrected.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCut** ([out, retval] VARIANT_BOOL* pbAutomaticCut)
HRESULT **put_AutomaticCut** ([in] VARIANT_BOOL bAutomaticCut)

The "AutomaticCut" property specifies if regions of high intensity caused by dust have been removed form the count rate data prior correlation and photon count histogram calculations.

pbAutomaticCut	[out]	A pointer to a variable which will receive the current state of the flag.

bAutomaticCut	[in]	The value 0 indicates that the high intensity regions have not been excluded. A different value indicates that the high intensity regions have not been excluded.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCutRatio** ([out, retval] double* pdRatio)
HRESULT **put_AutomaticCutRatio** ([in] double dRatio)

The "AutomaticCutRatio" property specifies the minimum intensity ratio of the high intensity region to the low intensity region where the automatic cut function excludes the high intensity ratio form the count rate data.

pdRatio	[out]	A pointer to a variable which will receive the cut intensity ratio.
dRatio	[in]	The cut intensity ratio.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorBinning** ([out, retval] double* pdBinning)
HRESULT **put_CorrelatorBinning** ([in] double dBinning)

The "CorrelatorBinning" property specifies the time for which the pulses are accumulated before they are passed to the correlator.

pdBinning	[out]	A pointer to a variable which will receive the correlator binning time in seconds.
dBinning	[in]	The correlator binning time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorMaximumTime** ([out, retval] double* pdMaximumTime)
HRESULT **put_CorrelatorMaximumTime** ([in] double dMaximumTime)

The "CorrelatorMaximumTime" property specifies the maximum correlation time which is considered for the correlation results. In addition, the maximum correlation time is limited by the measurement time.

pdMaximumTime	[out]	A pointer to a variable which will receive the maximum correlation time in seconds.
dMaximumTime	[in]	The maximum correlation time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorTauChannels** ([out, retval] long* plBinning)
HRESULT **put_CorrelatorTauChannels** ([in] long lBinning)

The "CorrelatorTauChannels" property specifies the number of channels that are generated per correlator tau-stage. The property determines the the density of the correlator channels.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCountRateBinnig**
([out, retval] VARIANT_BOOL* pbAutomatic)
HRESULT **put_AutomaticCountRateBinnig**
([in] VARIANT_BOOL bAutomatic)

The "AutomaticCountRateBinnig" property specifies if the count rate binning time for display in the count rate diagram is automatically ajusted by the online processing component during the acquisition. If not, the "CountRateBinnig" property is used instead.

pbAutomatic	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomatic	[in]	The value 0 indicates that the "CountRateBinnig" property is used. A different value indicates that that the binning time is determined automatically.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CountRateBinnig** ([out, retval] double* pdBinningTime)
HRESULT **put_CountRateBinnig** ([in] double dBinningTime)

The "CountRateBinnig" property specifies the time where count rate data are averaged for display in the count rate diagram when the automatic count rate binning is switched off (see "AutomaticCountRateBinnig").

pdBinningTime	[out]	A pointer to a variable which will receive the binning time in seconds.
dBinningTime	[in]	The binning time in seconds.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticPhotonCountHistogramBinnig**
([out, retval] VARIANT_BOOL* pbAutomatic)
HRESULT **put_AutomaticPhotonCountHistogramBinnig**
([in] VARIANT_BOOL bAutomatic)

The "AutomaticPhotonCountHistogramBinnig" property specifies if the photon count histogram binning time is automatically determined by the online processing component during the acquisition. If not, the "PhotonCountHistogramBinnig" property is used instead.

pbAutomatic	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomatic	[in]	The value 0 indicates that the "PhotonCountHistogramBinnig" property is used. A different value indicates that that the binning time is determined automatically.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PhotonCountHistogramBinnig** ([out, retval] double* pdBinningTime)
HRESULT **put_PhotonCountHistogramBinnig** ([in] double dBinningTime)

The "PhotonCountHistogramBinnig" property specifies the time where count rate data are accumulated for the generation of the photon count histogram when the automatic photon count histogram binning is switched off (see "AutomaticPhotonCountHistogramBinnig").

pdBinningTime	[out]	A pointer to a variable which will receive the binning time in seconds.
dBinningTime	[in]	The binning time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Dump** ([out,retval] BSTR* pState)

The "Dump" method generates a string with the informations about the current state of the acquisition parameters.

pState	[in]	A pointer to a variable which will receive the string with the acquisition parameter state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsSamplePositionParameters

The "IAimFcsSamplePositionParameters" interface is one of four interfaces for acquisition parameters. Via this interface the parameters for the the lateral and axial focus positions which are used for acquisition are accessible.

Methods

HRESULT **get_Name** ([out,retval] BSTR* ppName)
HRESULT **put_Name** ([in] BSTR pName)

The "Name" property specifies the name of the sample position data set which has been choosen by the user.

ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the sample position data set.
pName	[in]	A pointer to a character string with the name of the sample position data set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SamplePositionMode**
([out, retval] enumFcsSamplePositionMode* peMode)
HRESULT **put_SamplePositionMode**
([in] enumFcsSamplePositionMode eMode)

The "SamplePositionMode" property specifies which sample positions are used for data acquisition. This includes the decision between sample carrier and lsm position list and die order on a sample carrier.

peMode	[out]	A pointer to a variable which will receive the currently used sample position mode.
eMode	[in]	The identifier for the sample positions mode: eFcsSamplePositionModeCurrent - There is only one position an this is the current stage and focus position. eFcsSamplePositionModeList - The LSM position list is used. eFcsSamplePositionModeCarrierRows - The marked positions on the sample carrier are used in row by row order. eFcsSamplePositionModeCarrierColumns - The marked positions on the sample carrier are used in column by column order. eFcsSamplePositionModeCarrierRowsMeander - The marked positions on the sample carrier are used in row by row order with different direction for odd and even rows. eFcsSamplePositionModeCarrierColumnsMeander - The marked positions on the sample carrier are used in column by column order with different direction for odd and even columns. eFcsSamplePositionModeSequential - The LSM position list is used for fast scanner movement during acquisition based on the position list.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionListSize** ([out, retval] long* plSize)
HRESULT **put_PositionListSize** ([in] long lSize)

The "PositionListSize" property specifies the number of entries in the list of sample positions.

plSize	[out]	A pointer to a variable which will receive the current number of entries in the sample position list .
lSize	[in]	The new number of sample position list entries.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionX** ([in] long lIndex ,
[out, retval] double* pdX)
HRESULT **put_PositionX** ([in] long lIndex ,
[in] double dX)

The "PositionX" property specifies the x-stage coordinate of the sample position with the specified index in sample position list.

lIndex	[in]	The zero based sample position list index.
pdX	[out]	A pointer to a variable which will receive the used stage x-coordinate in metre.
dX	[in]	The new stage x-coordinate in metre.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionY** ([in] long lIndex ,
[out, retval] double* pdY)
HRESULT **put_PositionY** ([in] long lIndex ,
[in] double dY)

The "PositionY" property specifies the y-stage coordinate of the sample position with the specified index in sample position list.

lIndex	[in]	The zero based sample position list index.
pdY	[out]	A pointer to a variable which will receive the used stage y-coordinate in metre.
dY	[in]	The new stage y-coordinate in metre.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionZ** ([in] long lIndex ,
[out, retval] double* pdZ)
HRESULT **put_PositionZ** ([in] long lIndex ,

[in] double dZ)

The "PositionZ" property specifies the z-stage coordinate of the sample position with the specified index in sample position list.

lIndex	[in]	The zero based sample position list index.
pdZ	[out]	A pointer to a variable which will receive the used stage z-coordinate in metre.
dZ	[in]	The new stage z-coordinate in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierReferencePositionX** ([out, retval] double* pdReferenceX)
HRESULT **put_CarrierReferencePositionX** ([in] double dReferenceX)

The "CarrierReferencePositionX" property specifies the stage x-coordinate of the center of the left-top chamber.

pdReferenceX	[out]	A pointer to a variable which will receive the currently used x-coordinate of the center of the left-top chamber in metre.
dReferenceX	[in]	The new stage x-coordinate of the center of the left-top chamber in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierReferencePositionY** ([out, retval] double* pdReferenceY)
HRESULT **put_CarrierReferencePositionY** ([in] double dReferenceY)

The "CarrierReferencePositionY" property specifies the stage y-coordinate of the center of the left-top chamber.

pdReferenceY	[out]	A pointer to a variable which will receive the currently used y-coordinate of the center of the left-top chamber in metre.
dReferenceY	[in]	The new stage y-coordinate of the center of the left-top chamber in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierRows** ([out, retval] long* plRows)
HRESULT **put_CarrierRows** ([in] long lRows)

The "CarrierRows" property specifies the vertical number of chambers on the sample carrier.

plRows	[out]	A pointer to a variable which will receive the currently used vertical number of chambers.
lRows	[in]	The new vertical number of chambers.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierColumns** ([out, retval] long* plColumns)
HRESULT **put_CarrierColumns** ([in] long lColumns)

The "CarrierColumns" property specifies the horizontal number of chambers on the sample carrier.

plColumns	[out]	A pointer to a variable which will receive the currently used horizontal number of chambers.
lColumns	[in]	The new horizontal number of chambers.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierRowGap** ([out, retval] double* pdRowGap)
HRESULT **put_CarrierRowGap** ([in] double dRowGap)

The "CarrierRowGap" property specifies the vertical center to center distance of adjacent chamber on the sample carrier.

pdRowGap	[out]	A pointer to a variable which will receive the currently used vertical center to center chamber distance in metre.
dRowGap	[in]	The vertical center to center chamber distance in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierColumnGap** ([out, retval] double* pdColumnGap)
HRESULT **put_CarrierColumnGap** ([in] double dColumnGap)

The "CarrierColumnGap" property specifies the horizontal center to center distance of adjacent chamber on the sample carrier.

pdColumnGap	[out]	A pointer to a variable which will receive the currently used horizontal center to center chamber distance in metre.
dColumnGap	[in]	The horizontal center to center chamber distance in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierChamberUsed** ([in] long lRow ,
[in] long lColumn ,
[out, retval] VARIANT_BOOL* pbUsed)
HRESULT **put_CarrierChamberUsed** ([in] long lRow ,
[in] long lColumn ,

[in] VARIANT_BOOL bUsed)

The "CarrierChamberUsed" property specifies if data schould be acquired for a particular chamber.

lRow	[in]	The zero based index of the chamber row.
lColumn	[in]	The zero based index of the chamber column.
pbUsed	[out]	A pointer to a variable which will receive the current state of the chamber usage setting.
bUsed	[in]	A value different from zero indicates that data are acqired for the chamber and the value zero indicates that no data are acquired.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Dump** ([out,retval] BSTR* pState)

The "Dump" method generates a string with the informations about the current state of the acquisition parameters.

pState	[in]	A pointer to a variable which will receive the string with the acquisition parameter state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

ILockFiberOutBeampath

The "ILockFiberOutBeampath" inteface is used to sychronize the control of the hardware components with the LSM acquisition controller. Only one controller is allowed to control these components. The "SetSibling" method is called by an upper level component to inform the acquisition controller about the other controller. When an acquisition is started the fcs controller calls the "LockFiberOutBeampath" method of the other component to request the control of the shared hardware components.

Methods

HRESULT **LockFiberOutBeampath** ([in] VARIANT_BOOL bLock)

The "LockFiberOutBeampath" method is called by the LSM acquisition controller to lock or unlock the control of the shared hardware components.

bLock	[in]	The value 0 indicates that the control of the shared hardware is allowed. A different value indicates that the control of the shared hardware is forbidden.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **SetSibling** ([in] ILockFiberOutBeampath* pSibling)

The "SetSibling" method is called by an upper level component to inform the acquisition controller about the interface of a different controller for the hardware components shared whith the LSM part.

pSibling	[in]	A pointer to the interface of the other controller for the shared components or "NULL" if the other controller is no longer running.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

8.12. Component AimFcsData

IAimFcsDataBeamPath

The "IAimFcsDataBeamPath" interface provides acess to the beam path settings which have been used during acquisition.

Methods

HRESULT **Copy** ([in] IUnknown* pSource)

The "Copy" method copies the contents of the specified source beam path object to this object.

pSource	[in]	A pointer to one interface of the source beam path object. The object must have an IAimFcsDataBeamPath interface. The parameter can be set NULL to reset the object to the initial state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Name** ([out,retval] BSTR* ppName)
HRESULT **put_Name** ([in] BSTR pName)

The "Name" property specifies the name of the beam path which has been choosen by the user.

ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the beam path.
pName	[in]	A pointer to a character string with the name of the beam path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

STDMETHODIMP **get_NumberChannels** ([out,retval] long* plChannels)
STDMETHODIMP **put_NumberChannels** ([in] long lChannels)

The "NumberChannels" property specifies the number of detection channels which have been used to acquire the FCS data.

plChannels	[out]	A pointer to a variable which will receive the number of channels.
lChannels	[in]	The number of detection channels which have been used to acquire the FCS data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_ChannelIdentifier
                    ( [in] long                                  lChannel     ,
                      [out,retval] enumFcsBeamPathChannelIdentifier* peIdentifier )
HRESULT put_ChannelIdentifier
                    ( [in] long                                  lChannel     ,
                      [in] enumFcsBeamPathChannelIdentifier      eIdentifier  )
```

The "ChannelIdentifier" property specifies the unique identfier of a data channel which has been used during acquistion of the FCS data set.

lChannel	[in]	The zero based index of the channel.
peIdentifier	[out]	A pointer to a variable which will receive the identifier of the channel.
eIdentifier	[in]	The identifier of the channel: eFcsBeamPathChannelAutoCorrelation1 - auto correlation with data of frist detector. eFcsBeamPathChannelAutoCorrelation2 - auto correlation with data of second detector. eFcsBeamPathChannelCrossCorrelation1Versus2 - cross correlation with data of first detector versus second detector. eFcsBeamPathChannelCrossCorrelation2Versus1 - cross correlation with data of second detector versus first detector. eFcsBeamPathChannelInvalid - There is no such channel.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_ChannelName ( [in] long          lChannel ,
                          [out,retval] BSTR* ppName   )
HRESULT put_ChannelName ( [in] long          lChannel ,
                          [in] BSTR          pName    )
```

The "ChannelName" property specifies a user specified name for a data channel which has been used during acquistion of the FCS data set.

lChannel	[in]	The zero based index of the channel.
ppName	[out]	A pointer to a variable which will receive the user specified channel name.
pName	[in]	A pointer to a character string with the user specified channel name.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_NumberDetectors ( [out,retval] long* plDetectors )
HRESULT put_NumberDetectors ( [in] long          lDetectors  )
```

The "NumberDetectors" property specifies the number of detectors which have been used for data acquisition.

plDetectors	[out]	A pointer to a variable which will receive the number of unsed detectors.
lDetectors	[in]	The number of used detectors.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DetectorIdentifier**
([in] long lDetector ,
[out,retval] enumFcsBeamPathDetectorIdentifier* peIdentifier)
HRESULT **put_DetectorIdentifier**
([in] long lDetector ,
[in] enumFcsBeamPathDetectorIdentifier eIdentifier)

The "DetectorIdentifier" property specifies the unique identifier of a detector used during aqcusition.

lDetector	[in]	The zero based index of the detector.
peIdentifier	[out]	A pointer to a variable which will receive the identifier of the detector.
eIdentifier	[in]	The identifier of the detector: eFcsDetectorApd1 - The first avalache photo diode. eFcsDetectorApd2 - The second avalache photo diode. eFcsDetectorGaAsp1 - The first GaAsP detector. eFcsDetectorGaAsp2 - The second GaAsP detector. eFcsDetectorInvalid - There is no such detector.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DetectorVoltage** ([in] long lDetector ,
[out,retval] double* pdVoltage)
HRESULT **put_DetectorVoltage** ([in] long lDetector ,
[in] double dVoltage)

The "DetectorVoltage" property specifies the high voltage of an GaAsP detector which has been used during acquisition.

lDetector	[in]	The zero based index of the detector.
pdVoltage	[out]	A pointer to a variable which will receive the used voltage.
dVoltage	[in]	The voltage for the detector.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_NumberFilterSets** ([out,retval] long* plFilterSets)
HRESULT **put_NumberFilterSets** ([in] long lFilterSets)

The "NumberFilterSets" property specifies the number of filter sets in the acquisition beam path.

plFilterSets	[out]	A pointer to a variable which will receive the number of filter sets in the acquisition beam path.
lFilterSets	[in]	The number of filter sets in the acquisition beam path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_FilterSetIdentifier
                    (  [in] long                                  lFilterSet    ,
                       [out,retval] enumFcsBeamPathFilterSetIdentifier*  peIdentifier  )
HRESULT  put_FilterSetIdentifier
                    (  [in] long                                  lFilterSet    ,
                       [in] enumFcsBeamPathFilterSetIdentifier    eIdentifier   )
```

The "FilterSetIdentifier" property specifies the unique identifier of a filter sets used during aqcusition.

lFilterSet	[in]	The zero based index of the filter set.
peIdentifier	[out]	A pointer to a variable which will receive the identifier of the filter set.
eIdentifier	[in]	The identifier of the filter set: eFcsBeamPathFilterSetEmissionFilterDetector1 - the emission filter set next to the first detector. eFcsBeamPathFilterSetEmissionFilterDetector2 - the emission filter set next to the second detector. eFcsBeamPathFilterSetEmissionFilterDetector1_2 - the filter set where emmision light of the first and second detector transmits. eFcsBeamPathFilterSetMainBeamSplitter - the main beam splitter eFcsBeamPathFilterSetBeamSplitterDetectors1_2 - the dichroic beam splitter that seperates the light between FCS detector 1 and 2 eFcsBeamPathFilterSetLsmBeamSplitterDescanned1 eFcsBeamPathFilterSetLsmBeamSplitterDescanned2 eFcsBeamPathFilterSetLsmBeamSplitterDescanned3 - the LSM beam splitter in the FCS detection beam path. eFcsBeamPathFilterSetInvalid - there is no such filter set
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_Filter  (  [in] long              lFilterSet  ,
                        [out,retval] BSTR*     pFilter     )
HRESULT  put_Filter  (  [in] long              lFilterSet  ,
                        [in] BSTR              Filter      )
```

The "Filter" property specifies the name of the selected filter in a filter set used during aqcusition.

lFilterSet	[in]	The zero based index of the filter set.
pFilter	[out]	A pointer to a variable which will receive a pointer to the character string with the name of the filter.
Filter	[in]	A pointer to the character string with the name of the filter.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_NumberPinholes  (  [out,retval] long*  plPinholes  )
HRESULT  put_NumberPinholes  (  [in] long           lPinholes   )
```

The "NumberPinholes" property specifies the number of pinholes in the acquisition beam path.

plPinholes	[out]	A pointer to a variable which will receive the number of pinholes in the acquisition beam path.
lPinholes	[in]	The number of pinholes in the acquisition beam path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PinholeIdentifier**
([in] long lPinhole ,
[out,retval] enumFcsBeamPathPinholeIdentifier* peIdentifier)
HRESULT **put_PinholeIdentifier**
([in] long lPinhole ,
[in] enumFcsBeamPathPinholeIdentifier eIdentifier)

The "PinholeIdentifier" property specifies the unique identifier of a pinhole used during aqcusition.

lPinhole	[in]	The zero based index of the pinhole.
peIdentifier	[out]	A pointer to a variable which will receive the identifier of the pinhole.
eIdentifier	[in]	The identifier of the pinhole: eFcsBeamPathDetectorPinhole1 - a pinhole exclusive for detector 1 eFcsBeamPathDetectorPinhole2 - a pinhole exclusive for detector 2 eFcsBeamPathDetectorsPinhole1_2 - a pinhole for detector 1 and 2 eFcsBeamPathPinholeInvalid - there is no such pinhole
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PinholeDiameter** ([in] long lPinhole ,
[out,retval] double* pdDiameter)
HRESULT **put_PinholeDiameter** ([in] long lPinhole ,
[in] double dDiameter)

The "PinholeDiameter" property specifies the diameter of the pinhole specified by pinhole index that is used during acquisition.

lPinhole	[in]	The zero based index of the pinhole.
pdDiameter	[out]	A pointer to a variable which will receive the currently used pinhole diameter.
dDiameter	[in]	The new pinhole diameter in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AttenuatorArraySize** ([out,retval] long* plSize)
HRESULT **put_AttenuatorArraySize** ([in] long lSize)

The "AttenuatorArraySize" property specifies the number of attenuators which have been used

during acquisition.

plSize	[out]	A pointer to a variable which will receive the number of used attenuators in the acquisition beam path.
lSize	[in]	The number of used attenuators in the acquisition beam path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AttenuatorWavelength** ([in] long lAttenuator ,
[out,retval] double* pdWavelength)
HRESULT **put_AttenuatorWavelength** ([in] long lAttenuator ,
[in] double dWavelength)

The "AttenuatorWavelength" property specifies the wavelength of the attenuator specified by list index.

lAttenuator	[in]	The zero based index of the attenuator.
pdWavelength	[out]	A pointer to a variable which will receive the currently used wavelength in metre.
dWavelength	[in]	The new wavelength in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AttenuatorPower** ([in] long lAttenuator ,
[out,retval] double* pdPower)
HRESULT **put_AttenuatorPower** ([in] long lAttenuator ,
[in] double dPower)

The "AttenuatorPower" property specifies the relative power of the light that transmits the attenuator.

lAttenuator	[in]	The zero based index of the attenuator.
pdPower	[out]	A pointer to a variable which will receive the currently used relative power in the range 0..1.
dPower	[in]	The new relative power in the range 0..1.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AttenuatorArraySize** ([out,retval] long* plSize)
HRESULT **put_AttenuatorArraySize** ([in] long lSize)

The "BleachAttenuatorArraySize" property specifies the number of attenuators which have been used during the pre-bleach period of the acquisition.

plSize	[out]	A pointer to a variable which will receive the number of used attenuators in the acquisition beam path.

lSize	[in]	The number of used attenuators in the acquisition beam path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_BleachAttenuatorWavelength** ([in] long lAttenuator ,
[out,retval] double* pdWavelength)
HRESULT **put_BleachAttenuatorWavelength** ([in] long lAttenuator ,
[in] double dWavelength)

The "BleachAttenuatorWavelength" property specifies the wavelength of the attenuator specified by index in the list of attenuators used during the pre-bleach period of the acquisition.

lAttenuator	[in]	The zero based index of the attenuator.
pdWavelength	[out]	A pointer to a variable which will receive the currently used wavelength in metre.
dWavelength	[in]	The new wavelength in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_BleachAttenuatorPower** ([in] long lAttenuator ,
[out,retval] double* pdPower)
HRESULT **put_BleachAttenuatorPower** ([in] long lAttenuator ,
[in] double dPower)

The "BleachAttenuatorPower" property specifies the relative power of the light that transmitted the attenuator dring the pre-bleach period of the acquisition.

lAttenuator	[in]	The zero based index of the attenuator.
pdPower	[out]	A pointer to a variable which will receive the currently used relative power in the range 0..1.
dPower	[in]	The new relative power in the range 0..1.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_LaserArraySize** ([out,retval] long* plSize)
HRESULT **put_LaserArraySize** ([in] long lSize)

The "LaserArraySize" property specifies the number of lasers which have been used during acquisition.

plSize	[out]	A pointer to a variable which will receive the number of used lasers.
lSize	[in]	The number of used lasers.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT   get_LaserName  (  [in] long             lLaser    ,
                            [out,retval] BSTR*    ppName    )
HRESULT   put_LaserName  (  [in] long             lLaser    ,
                            [in] BSTR             pName     )
```

The "LaserName" property specifies the name of a laser which hase been used during acquisition.

lLaser	[in]	The zero based index of the laser.
ppName	[out]	A pointer to a variable which will receive the name of the laser.
pName	[in]	A pointer to a character string with the name of the laser.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT   get_LaserPower  (  [in] long             lLaser    ,
                            [out,retval] double*  pdPower   )
HRESULT   put_LaserPower  (  [in] long             lLaser    ,
                            [in] double           dPower    )
```

The "LaserPower" property specifies the laser power used during acquisition.

lLaser	[in]	The zero based index of the laser.
pdPower	[out]	A pointer to a variable which will receive the laser power in Watt.
dPower	[in]	The laser power in Watt.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT   get_LaserWavelengthArraySize  (  [in] long             lLaser    ,
                                           [out,retval] long*    plSize    )
HRESULT   put_LaserWavelengthArraySize  (  [in] long             lLaser    ,
                                           [in] long             lSize     )
```

The "LaserWavelengthArraySize" property specifies the number of lasers lines which have been used during acquisition for the laser specified by index in the list of used lasers.

lLaser	[in]	The zero based index of the laser.
plSize	[out]	A pointer to a variable which will receive the number of used laser lines.
lSize	[in]	The number of used laser lines.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT   get_LaserWavelength  (  [in] long             lLaser        ,
                                  [in] long             lIndex        ,
                                  [out,retval] double*  pdWavelength  )
HRESULT   put_LaserWavelength  (  [in] long             lLaser        ,
                                  [in] long             lIndex        ,
                                  [in] double           dWavelength   )
```

The "LaserWavelength" property specifies the wavelength of the laser line specified by laser line index which has been unsed during acquisition with the laser specified by laser index.

lLaser	[in]	The zero based index of the laser.
lIndex	[in]	The zero based index of the laser line.
pdWavelength	[out]	A pointer to a variable which will receive the laser line wavelength in metre.
dWavelength	[in]	The laser line wavelength in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetSizeForPersistSave** ([in] VARIANT_BOOL bText ,
[out,retval] long* plSize)

The serialization of all properties to a memory block must be done in two steps. "GetSizeForPersistSave" must be called first to obtain the size of the memory block that is required to receive all serialized properties. Then after a block of that size was allocated by the caller "PersistSave" must be called to transfer the serialized data to the buffer.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
plSize	[out]	Pointer to a variable to recive the size that is required for the serialized block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistSave** ([in] VARIANT_BOOL bText ,
[in] long lSize ,
[out,size_is(Size)] unsigned char* pData)

"PersistSave" serializes all properties to a memory block wich can be saved to a file. The caller should first request the size with "GetSizeForPersistSave" and then allocate a memory block of that size. Then "PersistSave" should be called to write the serialized data to the memory block.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
lSize	[in]	Size of the memory block. Not more than "lSize" bytes are copied from the serialized properties to the memory block "pData".
pData	[out]	Pointer to a memory block to receive the serialized data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistLoad** ([in] long lSize ,
[in] unsigned char* pData)

"PersistLoad" reconstructs the object properties from a memory block serialized with

"PersistSave".

lSize	[in]	Number of valid bytes in the memory block.
pData	[in]	Pointer to the memory block.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDataSamplePositions

The "IAimFcsDataAcquisitionParameters" interface provides access to the LSM sample position and sample carrier parameters which have been used during the acquisition of FCS data sets.

Methods

HRESULT **Copy** ([in] IUnknown* pSource)

The "Copy" method copies the contents of the specified source sample positions object to this object.

pSource	[in]	A pointer to one interface of the source sample positions object. The object must have an IAimFcsDataSamplePositions interface. The parameter can be set NULL to reset the object to the initial state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Name** ([out,retval] BSTR* ppName)
HRESULT **put_Name** ([in] BSTR pName)

The "Name" property specifies the name of the sample position data set which has been choosen by the user.

ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the sample position data set.
pName	[in]	A pointer to a character string with the name of the sample position data set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SamplePositionMode**
([out, retval] enumFcsDataSamplePositionMode* peMode)
HRESULT **put_SamplePositionMode**
([in] enumFcsDataSamplePositionMode eMode)

The "SamplePositionMode" property specifies which sample positions are used for data acquisition. This includes the decision between sample carrier and lsm position list and die order on a sample carrier.

peMode	[out]	A pointer to a variable which will receive the currently used sample position mode.

eMode	[in]	The identifier for the sample positions mode: eFcsDataSamplePositionModeCurrent - There is only one position an this is the current stage and focus position. eFcsDataSamplePositionModeList - The LSM position list is used. eFcsDataSamplePositionModeCarrierRows - The marked positions on the sample carrier are used in row by row order. eFcsDataSamplePositionModeCarrierColumns - The marked positions on the sample carrier are used in column by column order. eFcsDataSamplePositionModeCarrierRowsMeander - The marked positions on the sample carrier are used in row by row order with different direction for odd and even rows. eFcsDataSamplePositionModeCarrierColumnsMeander - The marked positions on the sample carrier are used in column by column order with different direction for odd and even columns. eFcsDataSamplePositionModeSequential - The LSM position list is used for fast scanner movement during acquisition based on the position list.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionListSize** ([out, retval] long* plSize)
HRESULT **put_PositionListSize** ([in] long lSize)

The "PositionListSize" property specifies the number of entries in the list of sample positions.

plSize	[out]	A pointer to a variable which will receive the current number of entries in the sample position list .
lSize	[in]	The new number of sample position list entries.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionX** ([in] long lIndex ,
[out, retval] double* pdX)
HRESULT **put_PositionX** ([in] long lIndex ,
[in] double dX)

The "PositionX" property specifies the x-stage coordinate of the sample position with the specified index in sample position list.

lIndex	[in]	The zero based sample position list index.
pdX	[out]	A pointer to a variable which will receive the used stage x-coordinate in metre.
dX	[in]	The new stage x-coordinate in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionY** ([in] long lIndex ,
[out, retval] double* pdY)
HRESULT **put_PositionY** ([in] long lIndex ,
[in] double dY)

The "PositionY" property specifies the y-stage coordinate of the sample position with the specified index in sample position list.

lIndex	[in]	The zero based sample position list index.
pdY	[out]	A pointer to a variable which will receive the used stage y-coordinate in metre.
dY	[in]	The new stage y-coordinate in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PositionZ** ([in] long lIndex ,
[out, retval] double* pdZ)
HRESULT **put_PositionZ** ([in] long lIndex ,
[in] double dZ)

The "PositionZ" property specifies the z-stage coordinate of the sample position with the specified index in sample position list.

lIndex	[in]	The zero based sample position list index.
pdZ	[out]	A pointer to a variable which will receive the used stage z-coordinate in metre.
dZ	[in]	The new stage z-coordinate in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierReferencePositionX** ([out, retval] double* pdReferenceX)
HRESULT **put_CarrierReferencePositionX** ([in] double dReferenceX)

The "CarrierReferencePositionX" property specifies the stage x-coordinate of the center of the left-top chamber.

pdReferenceX	[out]	A pointer to a variable which will receive the currently used x-coordinate of the center of the left-top chamber in metre.
dReferenceX	[in]	The new stage x-coordinate of the center of the left-top chamber in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierReferencePositionY** ([out, retval] double* pdReferenceY)
HRESULT **put_CarrierReferencePositionY** ([in] double dReferenceY)

The "CarrierReferencePositionY" property specifies the stage y-coordinate of the center of the left-top chamber.

pdReferenceY	[out]	A pointer to a variable which will receive the currently used y-coordinate of the center of the left-top chamber in metre.
dReferenceY	[in]	The new stage y-coordinate of the center of the left-top chamber in metre.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierRows** ([out, retval] long* plRows)
HRESULT **put_CarrierRows** ([in] long lRows)

The "CarrierRows" property specifies the vertical number of chambers on the sample carrier.

plRows	[out]	A pointer to a variable which will receive the currently used vertical number of chambers.
lRows	[in]	The new vertical number of chambers.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierColumns** ([out, retval] long* plColumns)
HRESULT **put_CarrierColumns** ([in] long lColumns)

The "CarrierColumns" property specifies the horizontal number of chambers on the sample carrier.

plColumns	[out]	A pointer to a variable which will receive the currently used horizontal number of chambers.
lColumns	[in]	The new horizontal number of chambers.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierRowGap** ([out, retval] double* pdRowGap)
HRESULT **put_CarrierRowGap** ([in] double dRowGap)

The "CarrierRowGap" property specifies the vertical center to center distance of adjacent chamber on the sample carrier.

pdRowGap	[out]	A pointer to a variable which will receive the currently used vertical center to center chamber distance in metre.
dRowGap	[in]	The vertical center to center chamber distance in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CarrierColumnGap** ([out, retval] double* pdColumnGap)
HRESULT **put_CarrierColumnGap** ([in] double dColumnGap)

The "CarrierColumnGap" property specifies the horizontal center to center distance of adjacent chamber on the sample carrier.

pdColumnGap	[out]	A pointer to a variable which will receive the currently used horizontal center to center chamber distance in metre.
dColumnGap	[in]	The horizontal center to center chamber distance in metre.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_CarrierChamberUsed ( [in] long                      lRow     ,
                                 [in] long                      lColumn  ,
                                 [out, retval] VARIANT_BOOL*    pbUsed   )
HRESULT put_CarrierChamberUsed ( [in] long                      lRow     ,
                                 [in] long                      lColumn  ,
                                 [in] VARIANT_BOOL              bUsed    )
```

The "CarrierChamberUsed" property specifies if data schould be acquired for a particular chamber.

lRow	[in]	The zero based index of the chamber row.
lColumn	[in]	The zero based index of the chamber column.
pbUsed	[out]	A pointer to a variable which will receive the current state of the chamber usage setting.
bUsed	[in]	A value different from zero indicates that data are acqired for the chamber and the value zero indicates that no data are acquired.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT CarrierPositionToRowAndColumn ( long  lPosition ,
                                        long* plRow     ,
                                        long* plColumn  )
```

The "CarrierPositionToRowAndColumn" method translates a measurment position index to the corresponding row and column indexes of the sample carrier.

lPosition	[in]	The zero based index of the position.
plColumn	[out]	The pointers to variables which will receive the carrier row and column indexes. If there is no such position or the current sample position mode is not applicable to a sample carrier the value -1 is written to both variables and S_OK is returned .
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT GetSizeForPersistSave ( [in] VARIANT_BOOL bText ,
                                [out,retval] long* plSize )
```

The serialization of all properties to a memory block must be done in two steps. "GetSizeForPersistSave" must be called first to obtain the size of the memory block that is required to receive all serialized properties. Then after a block of that size was allocated by the

caller "PersistSave" must be called to transfer the serialized data to the buffer.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
plSize	[out]	Pointer to a variable to recive the size that is required for the serialized block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistSave** ([in] VARIANT_BOOL bText ,
[in] long lSize ,
[out,size_is(Size)] unsigned char* pData)

"PersistSave" serializes all properties to a memory block wich can be saved to a file. The caller should first request the size with "GetSizeForPersistSave" and then allocate a memory block of that size. Then "PersistSave" should be called to write the serialized data to the memory block.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
lSize	[in]	Size of the memory block. Not more than "lSize" bytes are copied from the serialized properties to the memory block "pData".
pData	[out]	Pointer to a memory block to receive the serialized data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistLoad** ([in] long lSize ,
[in] unsigned char* pData)

"PersistLoad" reconstructs the object properties from a memory block serialized with "PersistSave".

lSize	[in]	Number of valid bytes in the memory block.
pData	[in]	Pointer to the memory block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDataFitParameters

The "IAimFcsDataFitParameters" interface provides access to to the parameters used for online and offline fit including fit results and statistics. The interface is used for remembering of the online fit parameters as acquisition parameters as well as for the oflline fit setup and result display.

Methods

HRESULT **Copy** ([in] IUnknown* pSource)

The "Copy" method copies the contents of the specified source fit parameters object to this object.

pSource	[in]	A pointer to one interface of the source fit parameters object. The object must have an IAimFcsDataFitParameters interface. The parameter can be set NULL to reset the object to the initial state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Name** ([out,retval] BSTR* ppName)
HRESULT **put_Name** ([in] BSTR pName)

The "Name" property specifies the name of the fit parameters data set which has been choosen by the user.

ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the fit parameters data set.
pName	[in]	A pointer to a character string with the name of the fit parameters data set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_FitRangeStart** ([out, retval] double* pdStart)
HRESULT **put_FitRangeStart** ([in] double dStart)

The "FitRangeStart" property specifies the lower limit for the range of the curve arguments which are considered for fitting.

pdStart	[out]	A pointer to a variable which will receive the lower limit of the fit range.
dStart	[in]	The lower limit of the fit range.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method

for the component can be used to obtain detailed error information.

HRESULT **get_FitRangeEnd** ([out, retval] double* pdEnd)
HRESULT **put_FitRangeEnd** ([in] double dEnd)

The "FitRangeEnd" property specifies the upper limit for the range of the curve arguments which are considered for fitting.

pdEnd	[out]	A pointer to a variable which will receive the upper limit of the fit range.
dEnd	[in]	The upper limit of the fit range.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_BasicModel** ([out, retval] enumFcsDataFitBasicModel* peModel)
HRESULT **put_BasicModel** ([in] enumFcsDataFitBasicModel eModel)

The "BasicModel" property specifies one of the three supported basic fitting models for correlation data or photon count histogram.

peModel	[out]	A pointer to a variable which will receive the identifier of the currently used basic fitting model.
eModel	[in]	The identifier of the fitting model: eFcsDataBasicModelCorrelation - The general model for correlation curves eFcsDataBasicModelPhotonCountHistogram - The general model for photon count histograms eFcsDataBasicModelUserFormula - A model specified by the user by a number of parameters to fit and a formula.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Formula** ([out, retval] BSTR* pFormula)
HRESULT **put_Formula** ([in] BSTR Formula)

The "Formula" property specifies the text with the user defined formula for the basic fitting model "eFcsDataBasicModelUserFormula".

pFormula	[out]	A pointer to a variable which will receive a pointer to a character string with the user specified formula.
Formula	[in]	A pointer to a character string with the user specified formula.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DimensionalityArraySize** ([out, retval] long* plDimensionalityArraySize)
HRESULT **put_DimensionalityArraySize** ([in] long lDimensionalityArraySize)

The "DimensionalityArraySize" property specifies the number of array elements in the array of the specification of the number of dimensions for the species in the sample. Note that it is not

neccessary to enlarge the array. The array is enlarged automatically when "put_Dimensionality" is called.

plDimensionalityArraySize	[out]	A pointer to a variable which will receive the current number of elements in the dimensionality array.
lDimensionalityArraySize	[in]	The new number of array elemets in the dimensionality array.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Dimensionality** ([in] long lComponent ,
[out, retval] long* plDimensionality)
HRESULT **put_Dimensionality** ([in] long lComponent ,
[in] long lDimensionality)

The "Dimensionality" property specifies the number of metric dimensions to consider for diffusion with enabled diffusion terms the specified molecule species for the basic model "eFcsDataBasicModelCorrelation" .

lComponent	[in]	The zero based index of the molecule species.
plDimensionality	[out]	A pointer to a variable which will receive the currently used number of dimensions.
lDimensionality	[in]	The number of dimensions: 1 - 1D, 2 - 2D or 3 - 3D.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ModelTerms** ([out, retval] long* plTerms)
HRESULT **put_ModelTerms** ([in] long lTerms)

The "ModelTerms" property specifies the model terms which have to be considered for the basic models "eFcsDataFitBasicModelCorrelation" and "eFcsDataFitBasicModelPhotonCountHistogram".

plTerms	[out]	A pointer to a variable which will receive the flags for the currently used model terms.
lTerms	[in]	A combination of flags which identify the terms of the basic model which are actually used. Possible flags are: For model eFcsDataFitBasicModelCorrelation: eFcsDataFitCorrelationFitTermOffset eFcsDataFitCorrelationFitTermAntiBunching eFcsDataFitCorrelationFitTermTripletState eFcsDataFitCorrelationFitTermBlinking eFcsDataFitCorrelationFitTermRotational eFcsDataFitCorrelationFitTermTranslational1 eFcsDataFitCorrelationFitTermTranslational2 eFcsDataFitCorrelationFitTermTranslational3 eFcsDataFitCorrelationFitTermFlow eFcsDataFitCorrelationFitTermKinetics
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method

for the component can be used to obtain detailed error information.

HRESULT **get_Terms** ([out, retval] long* plTerms)
HRESULT **put_Terms** ([in] long lTerms)

The "Terms" property specifies the terms which have choosen ba the user from the model terms "m_lModelTerms".

plTerms	[out]	A pointer to a variable which will receive the flags for the currently used model terms.
lTerms	[in]	A combination of flags which identify the terms of the basic model which are actually used. Possible flags are: For model eFcsDataFitBasicModelCorrelation: eFcsDataFitCorrelationFitTermOffset eFcsDataFitCorrelationFitTermAntiBunching eFcsDataFitCorrelationFitTermTripletState eFcsDataFitCorrelationFitTermBlinking eFcsDataFitCorrelationFitTermRotational eFcsDataFitCorrelationFitTermTranslational1 eFcsDataFitCorrelationFitTermTranslational2 eFcsDataFitCorrelationFitTermTranslational3 eFcsDataFitCorrelationFitTermFlow eFcsDataFitCorrelationFitTermKinetics
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Flags** ([out, retval] long* plFlags)
HRESULT **put_Flags** ([in] long lFlags)

The "Flags" property specifies additional flags for the selection of fit parameters from possible conversions for the basic models "eFcsDataFitBasicModelCorrelation" and "eFcsDataFitBasicModelPhotonCountHistogram".

plFlags	[out]	A pointer to a variable which will receive the currently used flags.
lFlags	[in]	A combination of flags which select the parameters to be fitted for some of the terms. Possible flags are: For model eFcsDataFitBasicModelCorrelation: - eFcsDataFitCorrelationFitTermFlagAnomal The anomaly parameters for the translational diffusion term have to be considered (fixed and not 1.0 or to be fitted). If the flag is not set the anomaly parameters are always 1.0. - eFcsDataFitCorrelationFitTermFlagDiffusionCoeff The diffusion coefficients or transport coefficients are model parameters rather than the diffusional correlation time. In this case the parameters - eFcsDataParam_D_1 - eFcsDataParam_D_2 - eFcsDataParam_D_3 - eFcsDataParam_omega_z and - eFcsDataParam_omega_r are used instead of - eFcsDataParam_tau_d_1 - eFcsDataParam_tau_d_2 - eFcsDataParam_tau_d_3 and - eFcsDataParam_S. - eCorrelationFitTermFlagFlowVelocity The velocity in the model parameters rather than the average residence time is used. In this case the parameter eFcsDataParam_v_f is used instead of eFcsDataParam_tau_f. - eFcsDataFitCorrelationFitTermFlagTwoPhoton Two photon excitation has been used for data acquisition.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_NumberParameters** ([out, retval] long* plNumberParameters)
HRESULT **put_NumberParameters** ([in] long lNumberParameters)

The "NumberParameters" property specifies the number entries in the list of fit parameters.

plNumberParameters	[out]	A pointer to a variable which will receive the currently used number of fit parameters.
lNumberParameters	[in]	The new number of fit parameters.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ParameterIdentifier** ([in] long lParameter ,
[out, retval] long* plIdentifier)
HRESULT **put_ParameterIdentifier** ([in] long lParameter ,
[in] long lIdentifier)

The "ParameterIdentifier" property specifies a unique identifier for the used individual fit parameters.

lParameter	[in]	The zero based index of the fit parameter.
plIdentifier	[out]	A pointer to a variable which will receive the indentifier of the parameter.
lIdentifier	[in]	The indentifier of the parameter. Possible values for the basic model "eFcsDataFitBasicModelCorrelation" are:
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ParameterTerm** ([in] long lParameter ,
[out,retval] BSTR* ppTerm)
HRESULT **put_ParameterTerm** ([in] long lParameter ,
[in] BSTR pTerm)

The "ParameterTerm" property specifies the name of the term in the formula where the parameter belongs. The term name is displayed in the user interface.

lParameter	[in]	The zero based index of the fit parameter.
ppTerm	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the term where the fit parameter belongs.
pTerm	[in]	A pointer to a character string with the name of the term where the fit parameter belongs.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ParameterName** ([in] long lParameter ,
[out,retval] BSTR* ppName)

```
HRESULT    put_ParameterName  (  [in] long          lParameter  ,
                                 [in] BSTR          pName       )
```

The "ParameterName" property specifies the name for the parameter, which is displayed in the user interface. The "ParameterName" property is used especially for the basic fit model "eFcsDataBasicModelUserFormula" where the name has to be derived form the formula.

lParameter	[in]	The zero based index of the fit parameter.
ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the fit parameter.
pName	[in]	A pointer to a character string with the name of the fit parameters data set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT    get_ParameterUnit  (  [in] long          lParameter  ,
                                 [out,retval] BSTR* ppUnit      )
HRESULT    put_ParameterUnit  (  [in] long          lParameter  ,
                                 [in] BSTR          pUnit       )
```

The "ParameterUnit" property specifies the name of the unit in which the start, minimum, maximum and result should be displayed in the user interface.

lParameter	[in]	The zero based index of the fit parameter.
ppUnit	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the unit.
pUnit	[in]	A pointer to a character string with the name of the unit.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT    get_ParameterUnitFactor  (  [in] long          lParameter  ,
                                      [out,retval] double* pdFactor    )
HRESULT    put_ParameterUnitFactor  (  [in] long          lParameter  ,
                                      [in] BSTR          pdFactor    )
```

The "ParameterUnitFactor" property specifies the factor that is applied to the start, minimum, maximum and result values to result in the displayed value for the unit with the in the "ParameterUnit" property.

lParameter	[in]	The zero based index of the fit parameter.
pdFactor	[out]	A pointer to a variable which will receive the unit factor.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT    get_ParameterPrecision  (  [in] long          lParameter  ,
                                     [out,retval] long* plCharacters )
HRESULT    put_ParameterPrecision  (  [in] long          lParameter  ,
                                     [in] long          lCharacters )
```

The "ParameterPrecision" property specifies the number of precision characters for the display for the parameter in the unit "ParameterUnit".

lParameter	[in]	The zero based index of the fit parameter.
plCharacters	[out]	A pointer to a variable which will receive the number of precision characters.
lCharacters	[in]	The number of precision characters.

HRESULT **get_ParameterMinimum** ([in] long lParameter ,
[out, retval] double* pdMinimum)
HRESULT **put_ParameterMinimum** ([in] long lParameter ,
[in] double dMinimum)

The "ParameterMinimum" property specifies the start of the value range for fit results for the specified parameter. The default value is "- DBL_MAX".

lParameter	[in]	The zero based index of the fit parameter.
pdMinimum	[out]	A pointer to a variable which will receive the minimum result value for the parameter.
dMinimum	[in]	The minimum result value for the parameter.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ParameterMaximum** ([in] long lParameter ,
[out, retval] double* pdMaximum)
HRESULT **put_ParameterMaximum** ([in] long lParameter ,
[in] double dMaximum)

The "ParameterMaximum" property specifies the end of the value range for fit results for the specified parameter. The default value is "DBL_MAX".

lParameter	[in]	The zero based index of the fit parameter.
pdMaximum	[out]	A pointer to a variable which will receive the maximum result value for the parameter.
dMaximum	[in]	The maximum result value for the parameter.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ParameterStartValue** ([in] long lParameter ,
[out, retval] double* pdStartValue)
HRESULT **put_ParameterStartValue** ([in] long lParameter ,
[in] double dStartValue)

The "ParameterStartValue" property specifies the start value for parameters of type "eFcsDataParameterTypeStartValue" and the fixed value for parameters of type

"eFcsDataParameterTypeFixed". The default value is 0.0.

lParameter	[in]	The zero based index of the fit parameter.
pdStartValue	[out]	A pointer to a variable which will receive the start value or fixed value for the parameter.
dStartValue	[in]	The start value or fixed value for the parameter.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ParameterLinkFlags** ([in] long lParameter ,
[out, retval] long* plLinkFlags)
HRESULT **put_ParameterLinkFlags** ([in] long lParameter ,
[in] long lLinkFlags)

The "ParameterLinkFlags" property specifies if the fit results for all channels, repetitions, kinetic indexes and/or sample positions should be linked for the fit parameter. In addition, there is a link index property which overwrites the fit parameters which are linked when this property is not less than zero.

lParameter	[in]	The zero based index of the fit parameter.
plLinkFlags	[out]	A pointer to a variable which will receive the currently used link flags.
lLinkFlags	[in]	A combination of the flags: eFcsDataParameterLinkFlagMeasurements - The parameter is linked for all measurements. eFcsDataParameterLinkFlagChannels - The parameter is linked for all channels. eFcsDataParameterLinkFlagRepeat - The parameter is linked for all repetitions. eFcsDataParameterLinkFlagKinetics - The parameter is linked for all kinetic indexes. eFcsDataParameterLinkFlagPosition - The parameter is linked for all sample positions.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ParameterLinkIndex** ([in] long lParameter ,
[out, retval] long* plLinkIndex)
HRESULT **put_ParameterLinkIndex** ([in] long lParameter ,
[in] long lLinkIndex)

The "ParameterLinkIndex" property can be used to specify that individual parameter of the same or different data sets are linked. A negative value indicates that the parameter is not linked individually and the "ParameterLinkFlags" property specifies if the parameter is still linked. All other values overwrite the settings in "ParameterLinkFlags".

lParameter	[in]	The zero based index of the fit parameter.
plLinkIndex	[out]	A pointer to a variable which will receive the currently used link index.
lLinkIndex	[in]	A negative number indicates that the parameter is not linked individually. Paraeters with the same non negativ value are linked.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_FitResultValid  ( [in] long                    lParameter  ,
                               [out, retval] VARIANT_BOOL*  pbValid     )
HRESULT  put_FitResultValid  ( [in] long                    lParameter  ,
                               [in] VARIANT_BOOL            bValid      )
```

The "FitResultValid" property specifies if the fit has been performed sucessfully and the value in the "FitResult" property is already valid.

lParameter	[in]	The zero based index of the fit parameter.
pbValid	[out]	A pointer to a variable which will receive the current state of the valid flag.
bValid	[in]	The value 0 indicates that the fit result value is not yet valid. All other values indicate that the fit result value is valid.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_FitResult  ( [in] long                 lParameter  ,
                          [out, retval] double*     pdFitResult )
HRESULT  put_FitResult  ( [in] long                 lParameter  ,
                          [in] double               dFitResult  )
```

The "FitResult" property specifies the result value of the fit for the property.

lParameter	[in]	The zero based index of the fit parameter.
pdFitResult	[out]	A pointer to a variable which will receive the fit result value.
dFitResult	[in]	The fit result value.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_FitResultStandardDeviation  ( [in] long          lParameter          ,
                                           [out, retval]      pdStandardDeviation )
                                           double*
HRESULT  put_FitResultStandardDeviation  ( [in] long          lParameter          ,
                                           [in] double        dStandardDeviation  )
```

The "FitResultStandardDeviation" property specifies the estimated standard deviation of the result value of the fit for the property.

lParameter	[in]	The zero based index of the fit parameter.
pdStandardDeviation	[out]	A pointer to a variable which will receive the fit result standard deviation.
dStandardDeviation	[in]	The fit result standard deviation.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_FitResultChiSquare  ( [out, retval] double*  pdChiSquare  )
```

HRESULT **put_FitResultChiSquare** ([in] double dChiSquare)

The "FitResultChiSquare" property specifies the quantity that is minimized by the fit routine for all parameters and linekd data sets at the time when the fit routine was complete.

pdChiSquare	[out]	A pointer to a variable which will receive the error.
dChiSquare	[in]	The fit error.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetSizeForPersistSave** ([in] VARIANT_BOOL bText ,
[out,retval] long* plSize)

The serialization of all properties to a memory block must be done in two steps. "GetSizeForPersistSave" must be called first to obtain the size of the memory block that is required to receive all serialized properties. Then after a block of that size was allocated by the caller "PersistSave" must be called to transfer the serialized data to the buffer.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
plSize	[out]	Pointer to a variable to recive the size that is required for the serialized block.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistSave** ([in] VARIANT_BOOL bText ,
[in] long lSize ,
[out,size_is(Size)] unsigned char* pData)

"PersistSave" serializes all properties to a memory block wich can be saved to a file. The caller should first request the size with "GetSizeForPersistSave" and then allocate a memory block of that size. Then "PersistSave" should be called to write the serialized data to the memory block.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
lSize	[in]	Size of the memory block. Not more than "lSize" bytes are copied from the serialized properties to the memory block "pData".
pData	[out]	Pointer to a memory block to receive the serialized data.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistLoad** ([in] long lSize ,
[in] unsigned char* pData)

"PersistLoad" reconstructs the object properties from a memory block serialized with "PersistSave".

lSize	[in]	Number of valid bytes in the memory block.
pData	[in]	Pointer to the memory block.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDataAcquisitionParameters

The "IAimFcsDataAcquisitionParameters" interface provides access to the parameters that have been used to acquire FCS-Data. The properties are used for display in the result window or for the re-use function to reconstruct acquisition parameters. The beam path, fit and sample position parameters are accessible via sub-interfaces.

Methods

HRESULT **Copy** ([in] IUnknown* pSource)

The "Copy" method copies the contents of the specified source acquisition parameters object to this object.

pSource	[in]	A pointer to one interface of the source acquisition parameters object. The object must have an IAimFcsDataAcquisitionParameters interface. The parameter can be set NULL to reset the object to the initial state.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Name** ([out,retval] BSTR* ppName)
HRESULT **put_Name** ([in] BSTR pName)

The "Name" property specifies the name of the acquisition parameter set which has been choosen by the user.

ppName	[out]	A pointer to a variable which will receive the pointer to the character string with the name of the acquisition parameter set.
pName	[in]	A pointer to a character string with the name of the acquisition parameter set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Comment** ([out,retval] BSTR* ppComment)
HRESULT **put_Comment** ([in] BSTR pComment)

The "Comment" property is a user specified text with additional informations about the acquisition parameter set.

ppComment	[out]	A pointer to a variable which will receive the pointer to the character string with the acquisition parameter set comment.

pComment	[in]	A pointer to a character string with the acquisition parameter set comment.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Category** ([out,retval] BSTR* ppCategory)
HRESULT **put_Category** ([in] BSTR pCategory)

The "Category" property is an additional text which can be used by the user to categorize acquisition parameter sets.

ppCategory	[out]	A pointer to a variable which will receive the pointer to the character string with the acquisition parameter set categorization text.
pCategory	[in]	A pointer to a character string with the acquisition parameter categorization text.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Status** ([out,retval] BSTR* ppStatus)
HRESULT **put_Status** ([in] BSTR pStatus)

The "Status" property is an additional text with informations about design state of the acquisition parameter set specified by the user.

ppStatus	[out]	A pointer to a variable which will receive the pointer to the character string with the design state of the acquisition parameter set
pStatus	[in]	A pointer to a character string with the design state of the acquisition parameter set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_User** ([out,retval] BSTR* ppUser)
HRESULT **put_User** ([in] BSTR pUser)

The "User" property specifies the login name of user who created the acquisition parameter set.

ppUser	[out]	A pointer to a variable which will receive the pointer to the character string with the name of user.
pUser	[in]	A pointer to a character string with the name of the name of user.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DateTime** ([out,retval] double* pdTime)

HRESULT **put_DateTime** ([in] double dTime)

The "DateTime" property specifies the date and time of the last modification of the aquisition parameter set.

pdTime	[out]	A pointer to a variable which will receive the time of the last modification of the aquisition parameter set.
dTime	[in]	The time of the last modification of the aquisition parameter set in days relative to December 30, 1899 0:00 a.m. The fractional part of the value represents the time of day.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_NumberChannels** ([out,retval] long* plChannels)
HRESULT **put_NumberChannels** ([in] long lChannels)

The "NumberChannels" property specifies the number of detection channels which have been used to acquire the FCS data.

plChannels	[out]	A pointer to a variable which will receive the number of channels.
lChannels	[in]	The number of detection channels which have been used to acquire the FCS data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ChannelIdentifier**
([in] long lChannel ,
[out,retval] enumFcsBeamPathChannelIdentifier* peIdentifier)
HRESULT **put_ChannelIdentifier**
([in] long lChannel ,
[in] enumFcsBeamPathChannelIdentifier eIdentifier)

The "ChannelIdentifier" property specifies the unique identfier of a data channel which ahs been used during acquistion of the FCS data set.

lChannel	[in]	The zero based index of the channel.
peIdentifier	[out]	A pointer to a variable which will receive the identifier of the channel: eFcsBeamPathChannelAutoCorrelation1 - auto correlation with data of frist detector. eFcsBeamPathChannelAutoCorrelation2 - auto correlation with data of second detector. eFcsBeamPathChannelCrossCorrelation1Versus2 - cross correlation with data of first detector versus second detector. eFcsBeamPathChannelCrossCorrelation2Versus1 - cross correlation with data of second detector versus first detector. eFcsBeamPathChannelInvalid - There is no such channel.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **FitParameters** ([in] long lChannel ,
[out,retval] IAimFcsDataFitParameters** ppFitParameters)

The "FitParameters" method yields the interface pointer for the object with the parameters which have been used for fitting during acquisition.

lChannel	[in]	The zero based index of the detection channel.
ppFitParameters	[out]	A pointer to a variable which will receive the interface pointer for the fit parameters object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **BeamPath** ([out,retval] IAimFcsDataBeamPath** ppBeamPath)

The "BeamPath" method yields the interface pointer for the object with the settings of the hardware objects in the acquisition beam path that have been used during acquisition.

ppBeamPath	[out]	A pointer to a variable which will receive the interface pointer for the beam path parameters object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **SamplePositions**
([out,retval] IAimFcsDataSamplePositions** ppSamplePositions)

The "SamplePositions" method yields the interface for the parameters for LSM sample positions and sample carrier which have been used during acquisition.

ppSamplePositions	[out]	A pointer to a variable which will receive the interface for the object with the sample position parameters.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_BleachTime** ([out, retval] double* pdBleachTime)
HRESULT **put_BleachTime** ([in] double dBleachTime)

The "BleachTime" property specifies the time the sample is illuminated prior the data acquisition. This bleaching is done for each sample position.

pdBleachTime	[out]	A pointer to a variable which will receive the currently used sample bleach time in seconds.
dBleachTime	[in]	The new sample bleach time in seconds.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MeasurementTime** ([out, retval] double* pdMeasurementTime)
HRESULT **put_MeasurementTime** ([in] double dMeasurementTime)

The "MeasurementTime" property specifies the duration of the data acquisiton per sample position, repeat index and kinetics index.

pdMeasurementTime	[out]	A pointer to a variable which will receive the currently used duration of the acqisition in seconds.
dMeasurementTime	[in]	The new duration of the acqisition in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MeasurementRepeat** ([out, retval] long* plMeasurementRepeat)
HRESULT **put_MeasurementRepeat** ([in] long lMeasurementRepeat)

The "MeasurementRepeat" property specifies the number of repetitions for the data acquisition at one sample position and kinetics index.

plMeasurementRepeat	[out]	A pointer to a variable which will receive the currently used number of repetitions.
lMeasurementRepeat	[in]	The new number of repetitions.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_KineticsCount** ([out, retval] long* plKineticsCount)
HRESULT **put_KineticsCount** ([in] long lKineticsCount)

The "KineticsCount" property specifies the number of acquistions at one sample position with delay for kinetic analysis.

plKineticsCount	[out]	A pointer to a variable which will receive the currently used number of aqcusitions.
lKineticsCount	[in]	The new number of aqcusitions.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_KineticsMeasurementStart** ([in] long lKineticIndex ,
[out, retval] double* pdStartTime)
HRESULT **put_KineticsMeasurementStart** ([in] long lKineticIndex ,

[in] double dStartTime)

The "KineticsMeasurementStart" property specifies the time of the start of the acquisition of the first data of an kinetics measurement relative to the end of the bleach time.

lKineticIndex	[in]	The zero based kinetic measurement index.
pdStartTime	[out]	A pointer to a variable which will receive the currently used acquistion start time.
dStartTime	[in]	The new acquistion start time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SequentialIIIumination**
([out, retval] VARIANT_BOOL* pbSequentialIIIumination)
HRESULT **put_SequentialIIIumination**
([in] VARIANT_BOOL bSequentialIIIumination)

The "SequentialIIIumination" property specifies if multi-channel acquisition is performed with sequential illumination to avoid blead through.

pbSequentialIIIumination	[out]	A pointer to a variable which will receive "VARIANT_TRUE" if the data are acquired in sequential mode and "VARIANT_FALSE" if the data are acquired in simultaneous mode.
bSequentialIIIumination	[in]	The value 0 indicates that the data for different illuminations are acquitred simultaneously. A different value indicates that they are acquired in sequential mode.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SequentialIIIuminationPeriod**
([out, retval] double* pdSequentialIIIuminationPeriod)
HRESULT **put_SequentialIIIuminationPeriod**
([in] double dSequentialIIIuminationPeriod)

The "SequentialIIIuminationPeriod" property specifies illumination period at one scanner position in "eFcsSamplePositionModeSequential" mode.

pdSequentialIIIuminationPeriod	[out]	A pointer to a variable which will receive the currently used illumination period in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SpotIlluminationPeriod** ([out, retval] double* pdSpotIlluminationPeriod)
HRESULT **put_SpotIlluminationPeriod** ([in] double dSpotIlluminationPeriod)

The "SpotIlluminationPeriod" property specifies illumination period at one scanner position in "eFcsSamplePositionModeSequential" mode.

pdSpotIlluminationPeriod	[out]	A pointer to a variable which will receive the currently used illumination period in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_BaseLineCorrection**
([out, retval] VARIANT_BOOL* pbBaseLineCorrection)
HRESULT **put_BaseLineCorrection**
([in] VARIANT_BOOL bBaseLineCorrection)

The "BaseLineCorrection" property specifies if the count rate inetensity drop due to bleaching has been corrected during acquisition.

pbBaseLineCorrection	[out]	A pointer to a variable which will receive the current state of the flag.
bBaseLineCorrection	[in]	The value 0 indicates that the count rate data have not been corrected. A different value indicates that the count rate data have been corrected.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCut** ([out, retval] VARIANT_BOOL* pbAutomaticCut)
HRESULT **put_AutomaticCut** ([in] VARIANT_BOOL bAutomaticCut)

The "AutomaticCut" property specifies if regions of high intensity caused by dust have been removed form the count rate data prior correlation and photon count histogram calculations.

pbAutomaticCut	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomaticCut	[in]	The value 0 indicates that the high intensity regions have not been excluded. A different value indicates that the high intensity regions have not been excluded.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCutRatio** ([out, retval] double* pdRatio)
HRESULT **put_AutomaticCutRatio** ([in] double dRatio)

The "AutomaticCutRatio" property specifies the minimum intensity ratio of the high intensity region

to the low intensity region where the automatic cut function excludes the high intensity ratio form the count rate data.

pdRatio	[out]	A pointer to a variable which will receive the cut intensity ratio.
dRatio	[in]	The cut intensity ratio.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorBinning** ([out, retval] double* dlBinning)
HRESULT **put_CorrelatorBinning** ([in] double dBinning)

The "CorrelatorBinning" property specifies the time for which the pulses are accumulated before they are passed to the correlator.

dBinning	[in]	The correlator binning time in seconds.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorMaximumTime** ([out, retval] double* pdMaximumTime)
HRESULT **put_CorrelatorMaximumTime** ([in] double dMaximumTime)

The "CorrelatorMaximumTime" property specifies the maximum correlation time which is considered for the correlation results. In addition, the maximum correlation time is limited by the measurement time.

pdMaximumTime	[out]	A pointer to a variable which will receive the maximum correlation time in seconds.
dMaximumTime	[in]	The maximum correlation time in seconds.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorTauChannels** ([out, retval] long* plBinning)
HRESULT **put_CorrelatorTauChannels** ([in] long lBinning)

The "CorrelatorTauChannels" property specifies the number of channels that are generated per correlator tau-stage. The property determines the the density of the correlator channels.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCountRateBinnig**
([out, retval] VARIANT_BOOL* pbAutomatic)
HRESULT **put_AutomaticCountRateBinnig**
([in] VARIANT_BOOL bAutomatic)

The "AutomaticCountRateBinnig" property specifies if the count rate binning time for display in the count rate diagram is automatically ajusted by the online processing component during the acquisition. If not, the "CountRateBinnig" property is used instead.

pbAutomatic	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomatic	[in]	The value 0 indicates that the "CountRateBinnig" property is used. A different value indicates that that the binning time is determined automatically.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CountRateBinnig** ([out, retval] double* pdBinningTime)
HRESULT **put_CountRateBinnig** ([in] double dBinningTime)

The "CountRateBinnig" property specifies the time where count rate data are averaged for display in the count rate diagram when the automatic count rate binning is switched off (see "AutomaticCountRateBinnig").

pdBinningTime	[out]	A pointer to a variable which will receive the binning time in seconds.
dBinningTime	[in]	The binning time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticPhotonCountHistogramBinnig**
([out, retval] VARIANT_BOOL* pbAutomatic)
HRESULT **put_AutomaticPhotonCountHistogramBinnig**
([in] VARIANT_BOOL bAutomatic)

The "AutomaticPhotonCountHistogramBinnig" property specifies if the photon count histogram binning time is automatically determined by the online processing component during the acquisition. If not, the "PhotonCountHistogramBinnig" property is used instead.

pbAutomatic	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomatic	[in]	The value 0 indicates that the "PhotonCountHistogramBinnig" property is used. A different value indicates that that the binning time is determined automatically.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PhotonCountHistogramBinnig** ([out, retval] double* pdBinningTime)
HRESULT **put_PhotonCountHistogramBinnig** ([in] double dBinningTime)

The "PhotonCountHistogramBinnig" property specifies the time where count rate data are

accumulated for the generation of the photon count histogram when the automatic photon count histogram binning is switched off (see "AutomaticPhotonCountHistogramBinnig").

pdBinningTime	[out]	A pointer to a variable which will receive the binning time in seconds.
dBinningTime	[in]	The binning time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AverageType**
([out, retval] enumFcsDataAverageType* peAverageType)
HRESULT **put_AverageType**
([in] enumFcsDataAverageType eAverageType)

The "AverageType" property specifies how the fit results of averaged data sets are calculated.

peAverageType	[out]	A pointer to a variable which will receive the identifier of the currently used averaging method.
eAverageType	[in]	An identifier which specifies the averaging method: eFcsDataAverageCorrelationResults The correlation curves are averaged and then fitted. eFcsDataAverageFitResults The fit results are averaged.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetSizeForPersistSave** ([in] VARIANT_BOOL bText ,
[out,retval] long* plSize)

The serialization of all properties to a memory block must be done in two steps. "GetSizeForPersistSave" must be called first to obtain the size of the memory block that is required to receive all serialized properties. Then after a block of that size was allocated by the caller "PersistSave" must be called to transfer the serialized data to the buffer.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
plSize	[out]	Pointer to a variable to recive the size that is required for the serialized block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistSave** ([in] VARIANT_BOOL bText ,
[in] long lSize ,
[out,size_is(Size)] unsigned char* pData)

"PersistSave" serializes all properties to a memory block wich can be saved to a file. The caller should first request the size with "GetSizeForPersistSave" and then allocate a memory block of that size. Then "PersistSave" should be called to write the serialized data to the memory block.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
lSize	[in]	Size of the memory block. Not more than "lSize" bytes are copied from the serialized properties to the memory block "pData".
pData	[out]	Pointer to a memory block to receive the serialized data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistLoad** ([in] long lSize ,
[in] unsigned char* pData)

"PersistLoad" reconstructs the object properties from a memory block serialized with "PersistSave".

lSize	[in]	Number of valid bytes in the memory block.
pData	[in]	Pointer to the memory block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsData

The "IAimFcsData" interface provides access to the data sets of a FCS measurement. The count rate, correlation, photon count histogram and count rate histogram data, file name of the raw data file and acquisition coordinates (channel, repetition, kinetics and sample position) are directly acessible via this interface. The measurement parameters and the fit parameters are acessible via sub-interfaces. The data sets are organized in a list where the individual informations are refrenced by the list index.

Methods

HRESULT **get_Name** ([out,retval] BSTR* pName)
HRESULT **put_Name** ([in] BSTR Name)

The "Name" property specifies the name of the data set list as specified by the user in a "Save" dialog box.

pName	[out]	A pointer to a variable to receive the a pointer to the character string with the data set list name.
Name	[in]	A pointer to a character string with the new name for the data set list.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Comment** ([out,retval] BSTR* pComment)
HRESULT **put_Comment** ([in] BSTR Comment)

The "Comment" property specifies a usually larger comment about the acquisition data set choosen by the user in a "Save" dialog box.

pComment	[out]	A pointer to a variable to receive the a pointer to the character string with the data set list comment.
Comment	[in]	A pointer to a character string with the new comment for the data set list.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DataSets** ([out,retval] long* plDataSets)
HRESULT **put_DataSets** ([in] long lDataSets)

The "DataSets" property specifies the number of entries in the list of FCS result data. Averageed/concatenated data sets are not considered.

plDataSets	[out]	A pointer to a variable which will receive the number of FCS result data sets.
lDataSets	[in]	The new number of FCS result data sets.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **DataSet** ([in] long lIndex ,
[out,retval] IAimFcsDataSet** ppDataSet)

The "DataSets" method yields the interface pointer for the object with the FCS result data specified by data set index.

lIndex	[in]	The zero based index of the data set. Averageed/ concatenated data sets are not considered.
ppDataSet	[out]	A pointer to a variable which will receive the interface pointer for the result data object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **CreateNew** ([in] long lDataSet)

The "CreateNew" method creates a new list entry at the specified index in the list of FCS result data.

lDataSet	[in]	The zero based index in the list where the new entry should be created.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Remove** ([in] long lDataSet)

The "Remove" method removes the entry specified by index from the list of FCS result data.

lDataSet	[in]	The zero based index of the list entry to remove.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Insert** ([in] long lIndex ,
[in] IAimFcsDataSet* pDataSet)

The "Insert" method inserts the nspecified interface to a FCS result data set in the list of FCS result data sets at the specified list index.

lIndex	[in]	The zero based index. Averageed/concatenated data sets are not considered.
pDataSet	[in]	A pointer to the interface of the FCS data set object to insert.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Copy** ([in] IUnknown* pSource)

The "Copy" method copies the contents of the specified source FCS result data object to this object.

pSource	[in]	A pointer to one interface of the source FCS result data object. The object must have an IAimFcsData interface. The parameter can be set NULL to erase the whole list.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetSizeForPersistSave** ([in] VARIANT_BOOL bText ,
[out,retval] long* plSize)

The serialization of all properties to a memory block must be done in two steps. "GetSizeForPersistSave" must be called first to obtain the size of the memory block that is required to receive all serialized properties. Then after a block of that size was allocated by the caller "PersistSave" must be called to transfer the serialized data to the buffer.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
plSize	[out]	Pointer to a variable to recive the size that is required for the serialized block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistSave** ([in] VARIANT_BOOL bText ,
[in] long lSize ,
[out,size_is(Size)] unsigned char* pData)

"PersistSave" serializes all properties to a memory block wich can be saved to a file. The caller should first request the size with "GetSizeForPersistSave" and then allocate a memory block of that size. Then "PersistSave" should be called to write the serialized data to the memory block.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
lSize	[in]	Size of the memory block. Not more than "lSize" bytes are copied from the serialized properties to the memory block "pData".
pData	[out]	Pointer to a memory block to receive the serialized data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistLoad** ([in] long lSize ,

[in] unsigned char* pData)

"PersistLoad" reconstructs the object properties from a memory block serialized with "PersistSave".

lSize	[in]	Number of valid bytes in the memory block.
pData	[in]	Pointer to the memory block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AverageFlags** ([out,retval] long* plAverageFlags)
HRESULT **put_AverageFlags** ([in] long lAverageFlags)

The "AverageFlags" property specifies which of the data sets are respected for the calculation of averaged and concatenated data sets.

plAverageFlags	[out]	A pointer to a variable which will receive the currently used flags.
lAverageFlags	[in]	A combination of the flags which identify the data sets to average: eFcsDataAverageFlagRepeat - all repetions eFcsDataAverageFlagKinetics - all kinetic idexes eFcsDataAverageFlagPosition - all sample positions eFcsDataAverageFlagChannels - all channels eFcsDataAverageFlagFitResults - The fit results are averaged rather than the correaltions. In all cases only the data sets with same measurement identifiers are averaged. When multiple flags are specified only one average data set is calculated for the combination of the specified data set coordinates.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SortOrder** ([out,retval] enumFcsDataSortOrder* peSortOrder)
HRESULT **put_SortOrder** ([in] enumFcsDataSortOrder eSortOrder)

The "SortOrder" property specifies the rule how the data sets are sorted. A general rule is that a reordering is done for data sets with the same measurement identifier only. The constants indicate how the data sets are sorted by their channel, repetition, sample position and kinetics.

peSortOrder	[out]	A pointer to a variable which will receive the indentifier of the currently used sort order.
eSortOrder	[in]	One of the following constants indicating the sort order: eFcsDataSortOrderChannelRepeatPositionKinetics - channel - repetition - sample position - kinetic eFcsDataSortOrderRepeatChannelPositionKinetics - repetition - channel - sample position - kinetic eFcsDataSortOrderRepeatPositionChannelKinetics - repetition - sample position - channel - kinetic eFcsDataSortOrderRepeatPositionKineticsChannel - repetition - sample position - kinetic - channel The first term indicates the index that is incremented first. When the last idex is reached the next trem indicates

the next index that is incremented.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_FitDataType** ([out,retval] enumFcsDataSortOrder* peDataType)
HRESULT **put_FitDataType** ([in] enumFcsDataSortOrder eDataType)

The "FitDataType" property specifies the type of data which is currently used for fitting.

peDataType [out] A pointer to a variable which will receive the currently used fit data type.
eDataType [in] The identifier of the fit data type: eFcsDataTypeCountRate - Count rate result data eFcsDataTypeCorrelation - Correlation result data eFcsDataTypePhotonCountHistogram - Photon count histogram result data eFcsDataTypePulseDistanceHistogram - Pulse distance histogram result data

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetNumberEntries** ([out,retval] long* plEntries)

The "GetNumberEntries" method can be used to determine the number entries in the sorted list with average and conatenation entries.

plEntries [out] A pointer to a variable which will receive the number of list entries.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Entry** ([in] long lIndex ,
[out,retval] IAimFcsDataSet** ppEntry)

The "Entry" method yields the interface pointer for the object with the FCS result data specified by data set index in the sorted list.

lIndex [in] The zero based index of the data set. Sorting and Averageed/concatenated data sets are considered.
ppEntry [out] A pointer to a variable which will receive the interface pointer for the result data object.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CurrentEntry** ([out,retval] long* plCurrentEntry)
HRESULT **put_CurrentEntry** ([in] long lCurrentEntry)

The "CurrentEntry" property specifies the index in the sorted with which is currently highlighted in the user interface.

plCurrentEntry	[out]	A pointer to a variable which will receive the current list index.
lCurrentEntry	[in]	The new current list index.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **EntryToDataSet** ([in] long lListEntry ,
[out,retval] long* plDataSet)

The "EntryToDataSet" method translates an index in the sorted list to the corresponding index in the pure data set list. If the translation result is -1 and the list entry exists the entry is an average/conatenated entry and the caller can use the "NumberDataSets" and "EntryDataSet" methods to determine the data sets to average or concatenate.

lListEntry	[in]	The index in the sorted list to translate.
plDataSet	[out]	A pointer to a variable which will receive the translated index (list of pure data sets).

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **DataSetToEntry** ([in] long lDataSet ,
[out,retval] long* plListEntry)

The "DataSetToEntry" method translates an index in the pure data set list to the corresponding index in the sorted list.

lDataSet	[in]	The index in the pure data sets list to translate.
plListEntry	[out]	A pointer to a variable which will receive the translated index (sorted list).

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **NumberDataSets** ([in] long lListEntry ,
[out,retval] long* plDataSets)

The "NumberDataSets" method can be used to dertermine the number of data sets which have to be averaged or concatenated for the specified entry in the sorted list.

lListEntry	[in]	The zero based index in the sorted list.
plDataSets	[out]	A pointer to a variable which will receive the number of data sets to average.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT EntryDataSet ( [in] long           lListEntry ,
                       [in] long           lIndex     ,
                       [out,retval] long*  plDataSet  )
```

The "EntryDataSet" method can be used to dertermine the indexes of data sets which have to be averaged or concatenated for the specified entry in the sorted list.

lListEntry	[in]	The zero based index in the sorted list.
lIndex	[in]	The zero based index in the list of data sets to average.
plDataSet	[out]	A pointer to a variable which will receive the the zero based index of the data set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT FindEntry ( [in] long                           lIdentifier1 ,
                    [in] long                           lIdentifier2 ,
                    [in] long                           lIdentifier3 ,
                    [in] long                           lIdentifier4 ,
                    [in] long                           lPosition    ,
                    [in] long                           lKinetics    ,
                    [in] long                           lRepeat      ,
                    [in] enumFcsBeamPathChannelIdentifier eChannel   ,
                    [out,retval] long*                  plEntry      )
```

The "FindEntry" searches the sorted list of data sets for the data set with the specified coordinates.lIdentifier1 lIdentifier2 lIdentifier3

lIdentifier4	[in]	The identifiers of the acquisition.
lPosition	[in]	The zero based index of the sample position in the list
lKinetics	[in]	The zero based kinetics measurement index.
lRepeat	[in]	The zero based index of the repetition.
eChannel	[in]	The identifier of the channel:
plEntry	[out]	A pointer to a varibale which will receive the zero based index of the data set with the specified coordinates. The value -1 is written to this variable if there is no such data set.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT Save ( [in] BSTR           pFile       ,
               [in] IUnknown*      pProgress   ,
               [in] long           lStartOffset ,
               [in] long           lType       ,
               [in] long           lDataSet    ,
               [in] long           lFileIndex  ,
               [in] VARIANT_BOOL   bRawData    ,
               [out,retval] long*  plEndOffset )
```

The "Save" method writes contensts of the data set list to a file.

pFile	[in]	A pointer to the character string with the name of the file.
pProgress	[in]	A pointer to one interface of the object which will receive the progress information. The object can also be used to abort the operation. The parameter can be NULL if no progress information is required.
lStartOffset	[in]	The offset in the file where the write operation should start.
lType	[in]	The identifier for the contents to write: eFcsFileLoadSaveTypeAll - All properties and data sets. eFcsFileLoadSaveTypeCurrent - All top level properties and the current data set. eFcsFileLoadSaveTypeSelected - All top level properties and the selected data sets. eFcsFileLoadSaveTypeHeading - The part prior the first data set eFcsFileLoadSaveTypeTail - The part after the last data set eFcsFileLoadSaveTypeEntry - A data set. The lDataSet parameter contains the index of the data set.
lDataSet	[in]	For the type eFcsFileLoadSaveTypeEntry this parameter is the zero based index of the data set to write. For other types the values has no meaning.
lFileIndex	[in]	For the type eFcsFileLoadSaveTypeEntry this parameter is the zero based index of the data set in the file. For other types the values has no meaning.
bRawData	[in]	The value zero indicates that the measurement data are written to the file only. A different value indicates that the files with the raw data are created in addition.
plEndOffset	[out]	A pointer to a variable which will receive the offset in the file after the block which has been written.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Load** (
[in] BSTR pFile ,
[in] IUnknown* pProgress ,
[in] long lStartOffset ,
[in] long lType ,
[in] long lDataSet ,
[out,retval] long* plEndOffset)

The "Load" method reads contensts of the data set list from a file.

pFile	[in]	A pointer to the character string with the name of the file.
pProgress	[in]	A pointer to one interface of the object which will receive the progress information. The object can also be used to abort the operation. The parameter can be NULL if no progress information is required.
lStartOffset	[in]	The offset in the file where the read operation should start.
lType	[in]	The identifier for the contents to read: eFcsFileLoadSaveTypeAll - All properties and data sets. eFcsFileLoadSaveTypeCurrent - Same as eFcsFileLoadSaveTypeAll. eFcsFileLoadSaveTypeSelected - Same as eFcsFileLoadSaveTypeAll. eFcsFileLoadSaveTypeHeading - The part prior the first data set eFcsFileLoadSaveTypeTail - The part after the last data set eFcsFileLoadSaveTypeEntry - The first data set. Subsequent values are used to identifiy the other data sets.
lDataSet	[in]	For the type eFcsFileLoadSaveTypeEntry this parameter is the zero based index of the data set to write. For other types the values has no meaning.
plEndOffset	[out]	A pointer to a variable which will receive the offset in the file after the block which has been read.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDataSet

The "IAimFcsData" interface provides access to the FCS result data a single FCS measurement for one channel. The count rate, correlation, photon count histogram and count rate histogram data, file name of the raw data file and acquisition coordinates (channel, repetition, kinetics and sample position) are directly acessible via this interface. The measurement parameters and the fit parameters are acessible via sub-interfaces.

Methods

HRESULT **Copy** ([in] IUnknown* pSource)

The "Copy" method copies the contents of the specified source data set to this object.

pSource	[in]	A pointer to one interface of the source FCS result data object. The object must have an "IAimFcsDataSet" interface.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Identifier1** ([out,retval] long* plIdentifier)
HRESULT **put_Identifier1** ([in] long lIdentifier)
HRESULT **get_Identifier2** ([out,retval] long* plIdentifier)
HRESULT **put_Identifier2** ([in] long lIdentifier)
HRESULT **get_Identifier3** ([out,retval] long* plIdentifier)
HRESULT **put_Identifier3** ([in] long lIdentifier)
HRESULT **get_Identifier4** ([out,retval] long* plIdentifier)
HRESULT **put_Identifier4** ([in] long lIdentifier)

The "Identifier..." properties specify 4 integer values which identify an acquisition. The identifier is generated during acquisition for all channel-, repeat-, kinetics and position indexes of the acquisition.

plIdentifier	[out]	A pointer to a variable which will receive the identification value.
lIdentifier	[in]	The new identification value.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_MeasurementIndex** ([out,retval] long* plMeasurementIndex)
HRESULT **put_MeasurementIndex** ([in] long lMeasurementIndex)

The "MeasurementIndex..." properties specify the index of the measurement in the data set list wehere the data saet belongs. All data sets of a data set list with the same measurement identifier have the same measurement index. Note that the measurement index is assigned ba the data set list object at run time and is not stored in the file. The data set object has this property for performance reasons only to prevent from searicng the data set list for equal identifiers.

plMeasurementIndex	[out]	A pointer to a variable which will receive the zero based mesaurement index.
lMeasurementIndex	[in]	The zero based mesaurement index.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Enabled** ([out,retval] VARIANT_BOOL* pbEnabled)
HRESULT **put_Enabled** ([in] VARIANT_BOOL bEnabled)

The "Enabled" property specifies if a the data set is considered for calculation of averaged data.

pbEnabled	[out]	A pointer to a variable which will receive the current state of the flag.
bEnabled	[in]	The value zero indicates that the data set is not considered. All other values indicate that the data set is considerd.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Selected** ([out,retval] VARIANT_BOOL* pbSelected)
HRESULT **put_Selected** ([in] VARIANT_BOOL bSelected)

The "Selected" property specifies if a data set is marked in the table of FCS result data sets for display of diagrams and list editing functions.

pbSelected	[out]	A pointer to a variable which will receive the current state of the flag.
bSelected	[in]	The value zero indicates that the data set is not marked. All other values indicate that the data set is marked.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AcquisitionTime** ([out,retval] double* pdTime)
HRESULT **put_AcquisitionTime** ([in] double dTime)

The "AcquisitionTime" property specifies the time and date of the aquisition of the data set.

pdTime	[out]	A pointer to a variable which will receive the time of the start of the data aquisition.
dTime	[in]	The time of the start of the data aquisition in days relative to December 30, 1899 0:00 a.m. The fractional part of the value represents the time of day.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method

for the component can be used to obtain detailed error information.

---

get_CreatedByUser ([out,retval] BSTR* pUser)
put_CreatedByUser ([in] BSTR User)

The "CreatedByUser" property specifies the OS login user name for the user who has aqcuired the data set.

pUser	[out]	A pointer to a variable to receive the a pointer to the character string with the user name.
User	[in]	A pointer to a character string with the new user name.
Returns		S_OK - success E_INVALIDARG - "NULL" pointer has been passed. E_OUTOFMEMORY - not enough mempry to copy the character string.

HRESULT **AcquisitionParameters**
([out,retval] IAimFcsDataAcquisitionParameters** ppAcquisitionParameters)

The "AcquisitionParameters" method yields the interface pointer for the object with the parameters that have been used for data acquisition.

ppAcquisitionParameters	[out]	A pointer to a variable which will receive the interface pointer for the acquisition parameters object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **FitParameters** ([out,retval] IAimFcsDataFitParameters** ppFitParameters)

The "FitParameters" method yields the interface pointer for the object with the parameters which are used for offline fitting.

ppFitParameters	[out]	A pointer to a variable which will receive the interface pointer for the fit parameters object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **FitDesign** ([out,retval] IAimFcsDataFitParameters** ppFitParameters)

The "FitDesign" method yields the interface pointer for the object with the parameters which are used for offline fitting during the design period.

ppFitParameters	[out]	A pointer to a variable which will receive the interface pointer for the fit design parameters object.
Returns		"S_OK" if successful or an error code. In the latter case the

"LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Position** ([out,retval] long* plPosition)
HRESULT **put_Position** ([in] long lPosition)

The "Position" property specifies the index of the sample position which has been used to acquire the data for the data set. The "Position", "Kinetics", "Repeat" and "ChannelIdentifier" properties identify the data set in the acquisition process with the acquistion identifiers "Identifier1" to "Identifier4".

plPosition	[out]	A pointer to a variable which will receive the zero based index of the sample position.
lPosition	[in]	The zero based index of the sample position in the list of LSM positions of on the sample carrier specified in the "AimFcsDataSamplePositions" object of the acquisition parameters.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Kinetics** ([out,retval] long* plKinetics)
HRESULT **put_Kinetics** ([in] long lKinetics)

The "Kinetics" property specifies the kinetics index where the data in the data set have been acquired. The "Position", "Kinetics", "Repeat" and "ChannelIdentifier" properties identify the data set in the acquisition process with the acquistion identifiers "Identifier1" to "Identifier4".

lKinetics	[in]	The zero based kinetics measurement index.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Repeat** ([out,retval] long* plRepeat)
HRESULT **put_Repeat** ([in] long lRepeat)

The "Repeat" property specifies the index of the repeatition where the data in the data set have been acquired. The "Position", "Kinetics", "Repeat" and "ChannelIdentifier" properties identify the data set in the acquisition process with the acquistion identifiers "Identifier1" to "Identifier4".

plRepeat	[out]	A pointer to a variable which will receive the zero based index of the repetition.
lRepeat	[in]	The zero based index of the repetition.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ChannelIdentifier**
([out,retval] enumFcsBeamPathChannelIdentifier* peIdentifier)
HRESULT **put_ChannelIdentifier**

([in] enumFcsBeamPathChannelIdentifier eIdentifier)

The "ChannelIdentifier" property specifies a value that identifies the channel which has been used to aquire the data for the data set. The "Position", "Kinetics", "Repeat" and "ChannelIdentifier" properties identify the data set in the acquisition process with the acquistion identifiers "Identifier1" to "Identifier4".

peIdentifier	[out]	A pointer to a variable which will receive the identifier of the channel.
eIdentifier	[in]	The identifier of the channel: eFcsBeamPathChannelAutoCorrelation1 - auto correlation with data of frist detector. eFcsBeamPathChannelAutoCorrelation2 - auto correlation with data of second detector. eFcsBeamPathChannelCrossCorrelation1Versus2 - cross correlation with data of first detector versus second detector. eFcsBeamPathChannelCrossCorrelation2Versus1 - cross correlation with data of second detector versus first detector. eFcsBeamPathChannelInvalid - The acquisition channel is unknown.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetChannelName** ([out,retval] BSTR* ppName)

The "MakeChannelName" method generates a character string for the name of a data channel which has been used during acquistion. If the user has specified an non empty name this name is used. Otherwise a name based on the type and detector(s) is geneated.

ppName	[out]	A pointer to a variable which will receive the channel name.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DataArraySize** ([in] enumFcsDataType eType ,
[out,retval] long* plArraySize)
HRESULT **put_DataArraySize** ([in] enumFcsDataType eType ,
[in] long lArraySize)

The "DataArraySize" property specifies the number of FCS result value pairs of the specified type.

eType	[in]	The identifier for the result data arrays: eFcsDataTypeCountRate - Count rate result data eFcsDataTypeCorrelation - Correlation result data eFcsDataTypePhotonCountHistogram - Photon count histogram result data eFcsDataTypePulseDistanceHistogram - Pulse distance histogram result data eFcsDataTypeCountRateCutRegions - Count rate cut regions
plArraySize	[out]	A pointer to a variable which will receive the size of the result arrays.
lArraySize	[in]	The new size of the result arrays.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetDataArray** ([in] enumFcsDataType eType ,
[in] long lArraySize ,
[out,size_is(ArraySize)] double* pdArray1 ,
[out,size_is(ArraySize)] double* pdArray2)

The "GetDataArray" method copies the contents of the FCS result data arrays of the specified type to caller provided buffers.

eType	[in]	The identifier for the result data arrays: eFcsDataTypeCountRate - Count rate result data with the time values in "pdArray1" and the average count rate in "pdArray2". eFcsDataTypeCorrelation - Correlation result data with the time values in "pdArray1" and the correlation values in "pdArray2". eFcsDataTypePhotonCountHistogram - Photon count histogram result data with the count rate in "pdArray1" and the frequency of the corresponding count rate in "pdArray2". eFcsDataTypePulseDistanceHistogram - Pulse distance histogram result data with the time difference between pulses in "pdArray1" and the frequency of the corresponding time differences in "pdArray2". eFcsDataTypeCountRateCutRegions - Count rate cut regions with the cut start time in "pdArray1" and the cut end time in "pdArray2".
lArraySize	[in]	The size of the result buffers "pdArray1" and "pdArray2".
pdArray2	[out]	The pointers to the buffers which will receive the FCS result data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **SetDataArray** ([in] enumFcsDataType eType ,
[in] long lArraySize ,
[in,size_is(ArraySize)] double* pdArray1 ,
[in,size_is(ArraySize)] double* pdArray2)

The "SetDataArray" method copies FCS result data from caller provided buffers to the result data arrays of the specified type.

eType	[in]	The identifier for the result data arrays: eFcsDataTypeCountRate - Count rate result data with the time values in "pdArray1" and the average count rate in "pdArray2". eFcsDataTypeCorrelation - Correlation result data with the time values in "pdArray1" and the correlation values in "pdArray2". eFcsDataTypePhotonCountHistogram - Photon count histogram result data with the count rate in "pdArray1" and the frequency of the corresponding count rate in "pdArray2". eFcsDataTypePulseDistanceHistogram - Pulse distance histogram result data with the time difference between pulses in "pdArray1" and the frequency of the corresponding time differences in "pdArray2". eFcsDataTypeCountRateCutRegions - Count rate cut regions with the cut start time in "pdArray1" and the cut end time in "pdArray2".
lArraySize	[in]	The number of array element with valid data in "pdArray1" and "pdArray2".
pdArray2	[out]	The pointers to the buffers with the FCS result data to copy.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetAverageCountRate** ([out] long* plAveragedValues ,
[out] double* pdAverageCountRate)

The "GetAverageCountRate" method calculates the average count rate for the count rate data. The cut region are not considered.

plAveragedValues	[out]	A pointer to a variable which will receive the number of values which have been averaged.
pdAverageCountRate	[out]	A pointer to a variable which will receive the average count rate in Hz.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetAverageCorrelation** ([in] double dStartTime ,
[in] double dEndTime ,
[out,retval] double* pdAverageCorrelation)

The "GetAverageCorrelation" method calculates the average correlation for for the specified correlation time region. If the region covers less than 4 correaltion values the average for the first 4 correlation values is calculated.dStartTime

dEndTime	[in]	The start and end correlation time of the correltion time region to consider.
pdAverageCorrelation	[out]	A pointer to a variable which will receive the average correltion.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetRawDataFileName** ([out] VARIANT_BOOL* pbTemporary ,
[out,retval] BSTR* ppFileName)
HRESULT **SetRawDataFileName** ([in] VARIANT_BOOL bTemporary ,
[in] BSTR pFileName)

The "RawDataFileName" property specifies the name of the file with the raw pulse differences. An empty string is used for the case that the raw data have not been recorded or are no longer present.

pbTemporary	[out]	A pointer a variable which will receive "VARINAT_TRUE" if the file is deleted when the object is destroyed and "VARIANT_FALSE" if not. The parameter can be NULL if the information is not requested.
ppFileName	[out]	A pointer to a variable which will receive a pointer to the character string with the file name.
bTemporary	[in]	A value different from zero indicates that the file should be deleted when the object is destroyed.
pFileName	[in]	A pointer to the character string with the new file name.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT GetMarkerFileName ( [out] VARIANT_BOOL* pbTemporary ,
                            [out,retval] BSTR*  ppFileName  )
HRESULT SetMarkerFileName ( [in] VARIANT_BOOL   bTemporary  ,
                            [in] BSTR           pFileName   )
```

The "MarkerFileName" property specifies the name of the file with the raw data cut markers. An empty string is used for the case that the raw data have not been recorded or are no longer present.

pbTemporary	[out]	A pointer a variable which will receive "VARINAT_TRUE" if the file is deleted when the object is destroyed and "VARIANT_FALSE" if not. The parameter can be NULL if the information is not requested.
ppFileName	[out]	A pointer to a variable which will receive a pointer to the character string with the file name.
bTemporary	[in]	A value different from zero indicates that the file should be deleted when the object is destroyed.
pFileName	[in]	A pointer to the character string with the new file name.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT get_DetectorFrequency1 ( [out,retval] double* pdFrequency )
HRESULT put_DetectorFrequency1 ( [in] double          dFrequency  )
HRESULT get_DetectorFrequency2 ( [out,retval] double* pdFrequency )
HRESULT put_DetectorFrequency2 ( [in] double          dFrequency  )
```

The "DetectorFrequency..." properties specifies the sampling frequency of the of the detectors which have been unsed for data acquisition. For data sets with channel type "eFcsBeamPathChannelAutoCorrelation" the "DetectorFrequency1" property specifies the frequency of the used detector. For data sets with channel type "eFcsBeamPathChannelCrossCorrelation" the "DetectorFrequency1" property specifies the frequency of the detector with the lower index and the "DetectorFrequency2" property specifies the frequency of the detector with the higher index.

pdFrequency	[out]	A pointer to a variable which will receive the detector frequency.
dFrequency	[in]	The detector frequency.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT GetSizeForPersistSave ( [in] VARIANT_BOOL bText ,
                                [out,retval] long* plSize )
```

The serialization of all properties to a memory block must be done in two steps. "GetSizeForPersistSave" must be called first to obtain the size of the memory block that is required to receive all serialized properties. Then after a block of that size was allocated by the caller "PersistSave" must be called to transfer the serialized data to the buffer.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.

plSize	[out]	Pointer to a variable to recive the size that is required for the serialized block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistSave** ([in] VARIANT_BOOL bText ,
[in] long lSize ,
[out,size_is(Size)] unsigned char* pData)

"PersistSave" serializes all properties to a memory block wich can be saved to a file. The caller should first request the size with "GetSizeForPersistSave" and then allocate a memory block of that size. Then "PersistSave" should be called to write the serialized data to the memory block.

bText	[out]	A flag that indicates if the memory block should be generated in text format. If the flag is not set the data balock is gnerated in raw format.
lSize	[in]	Size of the memory block. Not more than "lSize" bytes are copied from the serialized properties to the memory block "pData".
pData	[out]	Pointer to a memory block to receive the serialized data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PersistLoad** ([in] long lSize ,
[in] unsigned char* pData)

"PersistLoad" reconstructs the object properties from a memory block serialized with "PersistSave".

lSize	[in]	Number of valid bytes in the memory block.
pData	[in]	Pointer to the memory block.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimProgressFifo

The "IAimProgressFifo" interface encapsulates a list of progress information interfaces intended for progress display for the status of running operation in progress bars. The "Append" method appends a operation to the tail of the fifo. The "Get" method returns the interface pointer for the object at the head. The "Remove" method removes the head element.

8.13. Component AimFcsFile

IAimFcsFileRead

The "IAimFcsFileRead" interface provides methods for file read operations of FCS data files in several formats. The general procedure is: 1. Specify the names of the files to read. 2. Determine the format of the individusl files with the "Format" method. 3. Create the destination FCS data objects and specify them with "put_Destination". It is possible to specify the same destination object for multiple files. The data sets are always appended to the FCS data object. It is recommended to use the results of the "Format" method to determine when file contents should be written to the same FCS data object. 4. Call "Run" to read a single file or all files. The "Start" method can be used instead of the "Run" method when the file read operation should be performed in a different program thread. The progress interface of the object can be used to determine the file read progress or to abort the operation.

Methods

HRESULT **get_NumberFiles** ([out,retval] long* plNumberFiles)
HRESULT **put_NumberFiles** ([in] long lNumberFiles)

The "NumberFiles" property specifies the number of entries in the list of source files for the file read operation. Note that it is not neccessary to call "put_NumberFiles" to enlarge the list for new file names. The list is enlarged automatically when new file names are sepecified with "put_FileName".

plNumberFiles	[out]	A pointer to a variable which will receive the current number of entries in the file list.
lNumberFiles	[in]	The new number of entries in the file list.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_FileName** ([in] long lIndex ,
[out,retval] BSTR* ppFileName)
HRESULT **put_FileName** ([in] long lIndex ,
[in] BSTR pFileName)

The "FileName" property specifies the name of a source file for the specified entry in the list of source files. For new list entries the list is enlarged automatically.

lIndex	[in]	The zero based index of the entry in the source file name list.
pFileName	[in]	A pointer to a character string with the name of the file.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_Destination ( [in] long                lIndex        ,
                           [out,retval] IUnknown**  ppDestination )
HRESULT  put_Destination ( [in] long                lIndex        ,
                           [in] IUnknown*           pDestination  )
```

The "Destination" property specifies the interface to the destination FCS data object for the specified entry in the list of source files. For new list entries the list is enlarged automatically.

lIndex	[in]	The zero based index of the entry in the source file name list.
ppDestination	[out]	A pointer to a variable which will receive the pointer to the inteface of the destination FCS data object.
pDestination	[in]	A pointer to one inteface of the destination FCS data object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  get_Correlator ( [out,retval] IUnknown**  ppCorrelator )
HRESULT  put_Correlator ( [out,retval] IUnknown*   pCorrelator  )
```

The "Correlator" property specifies the pointer to one interface of the correlator object which should be used to correlate data from raw data files. The common correlator properties for all challes should have been initialized and aure used. The channel properties are set up by the file read object. If the "Correlator" property is "NULL" the file read object will create an own correlator an will use the correlator default properties.

ppCorrelator	[out]	A pointer to a variable which will receive the pointer to the inteface of the correlator object.
pCorrelator	[in]	A pointer to one inteface of the correlator object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

```
HRESULT  GetFormat ( [in] long                          lIndex    ,
                     [out,retval] enumFcsFileFormat*    peFormat  )
```

The "GetFormat" method reads the header of the file specified by list index and determines the format of the file.

lIndex	[in]	The zero based index of the entry in the source file name list.
peFormat	[out]	A pointer to a variable which will receive the identifier of the file format: eFcsFileFormatConfoCor1 - ConfoCor1 measurement data file format eFcsFileFormatConfoCor2 - ConfoCor2 measurement data file format eFcsFileFormatConfoCor3 - ConfoCor3 measurement data file format eFcsFileFormatRawConfoCor2 - ConfoCor2 raw data file format eFcsFileFormatRawConfoCor3 - ConfoCor3 raw data file format

	eFcsFileFormatConfoCor3WithRawData - ConfoCor3 measurement data file format and raw data.
Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsFileWrite

The "IAimFcsFileWrite" interface provides methods for file write operation of FCS data files in several formats. The general procedure is:1. Specify the name, format and source FCS data object. 2. Specify a file write type which selects the data sets to write form the FCS data object. 3. Call "Run" to write the file(s). The "Start" method can be used instead of the "Run" method when the file write operation should be performed in a different program thread. The progress interface of the object can be used to determine the file write progress or to abort the operation.

Methods

HRESULT **get_FileName** ([out,retval] BSTR* ppFileName)
HRESULT **put_FileName** ([in] BSTR pFileName)

The "FileName" property specifies the name of the destination file for the file write operation.

pFileName	[in]	A pointer to a character string with the name of the file.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Format** ([out,retval] enumFcsFileFormat* peFormat)
HRESULT **put_Format** ([in] enumFcsFileFormat eFormat)

The "Format" property specifies the format of the file to write.

peFormat	[out]	A pointer to a variable which will receive the identifier of the file format.
eFormat	[in]	The identifier of the file format: eFcsFileFormatConfoCor1 - ConfoCor1 measurement data file format eFcsFileFormatConfoCor2 - ConfoCor2 measurement data file format eFcsFileFormatConfoCor3 - ConfoCor3 measurement data file format eFcsFileFormatRawConfoCor2 - ConfoCor2 raw data file format eFcsFileFormatRawConfoCor3 - ConfoCor3 raw data file format eFcsFileFormatConfoCor3WithRawData - ConfoCor3 measurement data file format and raw data.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Source** ([out,retval] IUnknown** ppSource)
HRESULT **put_Source** ([in] IUnknown* pSource)

The "Source" property specifies the interface to the source FCS data object for the file write operation.

ppSource	[out]	A pointer to a variable which will receive the pointer to the inteface of the source FCS data object.
pSource	[in]	A pointer to one inteface of the source FCS data object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_FileWriteType** ([out,retval] enumFcsFileWriteType* peType)
HRESULT **put_FileWriteType** ([in] enumFcsFileWriteType eType)

The "FileWriteType" property specifies how the data sets which are respected for the file write operation are selected.

peType	[out]	A pointer to a variable which will receive the identfier for the data set selection criterion.
eType	[in]	The identfier for the data set selection criterion: eFcsFileWriteTypeAll - All data sets have to be written. eFcsFileWriteTypeSelected - The selected data sets have to be written. If there is no selected data set then all data sets have to be written. eFcsFileWriteTypeCurrent - The current data set has to be written. If there is no current data set then all data sets have to be written.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **DestinationFilesExist** ([out] BSTR* ppFilesThatExist ,
[out,retval] VARIANT_BOOL* pbExist)

The "DestinationFilesExist" methods determines if one or multiple destination files already exist and assembles a list of the names of these files.

ppFilesThatExist	[out]	A pointer to a variable which will receive a pointer to the character string with the names of the files which exist. The file names are separated by '\n' characters.
pbExist	[out]	A pointer to a variable which will receive VARIANT_TRUE if there is at leat one file which exits and VARIANT_FALSE if not.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsMethodItemList

The "IAimFcsMethodItemList" interface provides methods for access to the contents of files with acquisition parameters, beam path settings and sample carrier position properties.

8.14. Component AimFcsProcessing

IAimFcsProcessingFit

The "IAimFcsProcessingFit" interface provides access to the fit routines for one or multiple data sets. Additional methods can be used to filter the actually used fit parameters, to calculate threorethical curves and to obtain fit parameter state informations during the during fit design.

Methods

HRESULT **FitRun** ([in] IUnknown* pDataSets ,
[in] IUnknown* pDataSet ,
[in] long lFlags)

The "FitRun" method performes the fit for one or multiple data sets in the specified data set list. The fit parameters are expected in the design fit parameters objects of the individual data sets. The results are written to the fit parameters objects of the individual data sets.

pDataSets	[in]	An interface pointer of the data set list object.
pDataSet	[in]	A pointer to the data set for which the fit is performed. When the "eAimFcsProcessingFitRunDeriveParameters" flag is set the fit parameters if this data set are also used for the other fitted data sets.
lFlags	[in]	The flags for the selection of the data sets to fit: eAimFcsProcessingFitRunDeriveParameters - The parameters of the specified data set are used for all fitted data sets. eAimFcsProcessingFitRunAllMeasurements - The fit is performed for all measurements eAimFcsProcessingFitRunAllChannels - The fit is performed for all channels eAimFcsProcessingFitRunAllRepetitions - The fit is performed for all repetitions eAimFcsProcessingFitRunAllKinetics - The fit is performed for all kinetics indexes eAimFcsProcessingFitRunAllPositions - The fit is performed for all sample positions eAimFcsProcessingFitRunAllSelected - The fit is performed for all selected data sets eAimFcsProcessingFitRunAll - The fit is performed for all data sets
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **FitStart** ([in] IUnknown* pDataSets ,
[in] IUnknown* pDataSet ,
[in] long lFlags)

The "FitStart" method satrts a thread which performes the fit for one or multiple data sets in the specified data set list. The fit parameters are expected in the design fit parameters objects of the individual data sets. The results are written to the fit parameters objects of the individual data sets.

pDataSets	[in]	An interface pointer of the data set list object.

pDataSet	[in]	A pointer to the data set for which the fit is performed. When the "eAimFcsProcessingFitRunDeriveParameters" flag is set the fit parameters if this data set are also used for the other fitted data sets.
lFlags	[in]	The flags for the selection of the data sets to fit: eAimFcsProcessingFitRunDeriveParameters - The parameters of the specified data set are used for all fitted data sets. eAimFcsProcessingFitRunAllMeasurements - The fit is performed for all measurements eAimFcsProcessingFitRunAllChannels - The fit is performed for all channels eAimFcsProcessingFitRunAllRepetitions - The fit is performed for all repetitions eAimFcsProcessingFitRunAllKinetics - The fit is performed for all kinetics indexes eAimFcsProcessingFitRunAllPositions - The fit is performed for all sample positions eAimFcsProcessingFitRunAllSelected - The fit is performed for all selected data sets eAimFcsProcessingFitRunAll - The fit is performed for all data sets
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **CalculateTheorethicalCurve**

```
( [in] IUnknown*                          pFitResults   ,
  [in] long                               lArguments    ,
  [in,size_is(lArguments)] double*        pdArguments   ,
  [out,size_is(Arguments)] double*        pdResults     )
```

The "CalculateTheorethicalCurve" method calculates a theorethical curve for basic model, terms and fit result values form the specified fit parameter and result object.

pFitResults	[in]	A pointer to one interface of the object with the fit parameters and results. The object must have an "IAimFcsDataFitParameters" interface.
lArguments	[in]	The number of curve points (function argument array and result array size).
pdArguments	[in]	The pointer to an array with the function arguments.
pdResults	[out]	The pointer to the array which will receive the function results.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **ParameterState**

```
( [in] IUnknown*                                                pDataSets   ,
  [in] IUnknown*                                                pDataSet    ,
  [out] BSTR*                                                   pMessage    ,
  [in] long                                                     lParameters ,
  [out,size_is(Parameters)] enumAimFcsProcessingFitParameterState* peState   )
```

The "ParameterState" method determines the state message for a fit with the design fit parameters of the specified data set. The method also generates a list with the state identifier for all fit parameters.

pDataSets	[in]	An interface pointer of the data set list object.
pDataSet	[in]	An interface pointer of the data set for which the check is performed.
pMessage	[out]	A pointer to a variable which will receive the character string with the

		state message.
lParameters	[in]	The number of arrea elements in the buffer "peState".
peState	[out]	A pointer to the array which will receive the state identifier for the individual parameters. The array index is the zero based index of the parameter in the design fit parameters object of the data set. Possible identifiers are: eAimFcsProcessingFitParameterStateOk - No problem for the fit module eAimFcsProcessingFitParameterStateProblem - The setting for this parameter or one of the other parameters of the same state should be changed to avoid instability of the fit results. eAimFcsProcessingFitParameterStateError - The setting for this parameter or one of the other parameters of the same state caused the impossibility to fit the data sets.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **InitializeFitParameters** ([in] IUnknown* pDestination ,
[in] IUnknown* pParameters ,
[in] IUnknown* pModel)

The "InitializeFitParameters" method initializes a fit parameter object form a fit parameter object with the model properties and a second fit parameter object with the properties of the parameters.

pDestination	[in]	The interface pointer for the destination fit parameter object.
pParameters	[in]	An interface pointer for the object with the properties of the model parameters.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **NumberSupportedBasicModels** ([out,retval] long* plModels)

The "NumberSupportedBasicModels" returns the number basic models which are supported by the fit module.

plModels	[out]	A pointer to a variable which will receive the number of basic models.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **AllModelParameters** ([in] long lModel ,
[in] IUnknown* pPotentialParameters)

The "AllModelParameters" method fills a "AimFitParameters" object with all informations about the supported model specified by index including all supported fit parameters, names and default values. Note that for the basic model "eFcsDataFitBasicModelUserFormula" the list of parameters remains empty.

lModel	[in]	The zero based index of the supported model.
pPotentialParameters	[in]	A pointer to the "AimFitParameters" object which will receive the moedl properties. The object must have an "IAimFitParameters" interface.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **ParseFormula** ([in] BSTR pFormula ,
[out,retval] BSTR* ppOperations)

The "ParseFormula" method parses the specified formula and assembles a text with the operations which are performed when the formula is used for fitting.

pFormula	[in]	A pointer to the character string with the formula.
ppOperations	[out]	A pointer to a variable which will receive a pointer to the character string with the list of operations.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsProcessingCorrelator

The "IAimFcsProcessingCorrelator" interface provides access to the claculation routines for correlation, count rate, photon count histogram and pulse distance histogram.Most of the properties influence the size of the destination arrays and the units in which the data are calculated. The other properties "m_Destination" and "DestinationFlags" specify the destination data sets and the type of result data which have to be written to the data set. Note that the "ChannelIdentifier" proerty of the data set and the "DetectorFrequency1" "DetectorFrequency1" and "DetectorFrequency2" properties of the acquisition paraeter information of the data sets are also used by the correlation module.The properties have to be set up before the calculation is statyed with the "Start" method. The caller should then use the "AppendRawData" method to specify the acquired raw data in the order in which they have been acquired. When all raw data have been passed the "Stop" method should be called to write the final results to the destionation data sets. Note that the destionation data sets are also updated priodically when new raw data are appended.

Methods

HRESULT **get_AutomaticCut** ([out, retval] VARIANT_BOOL* pbAutomaticCut)
HRESULT **put_AutomaticCut** ([in] VARIANT_BOOL bAutomaticCut)

The "AutomaticCut" property specifies if regions of high intensity caused by dust have been removed form the count rate data prior correlation and photon count histogram calculations.

pbAutomaticCut	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomaticCut	[in]	The value 0 indicates that the high intensity regions have not been excluded. A different value indicates that the high intensity regions have not been excluded.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCutRatio** ([out, retval] double* pdRatio)
HRESULT **put_AutomaticCutRatio** ([in] double dRatio)

The "AutomaticCutRatio" property specifies the minimum intensity ratio of the high intensity region to the low intensity region where the automatic cut function excludes the high intensity ratio form the count rate data.

pdRatio	[out]	A pointer to a variable which will receive the cut intensity ratio.

dRatio	[in]	The cut intensity ratio.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorBinning** ([out, retval] double* pdBinning)
HRESULT **put_CorrelatorBinning** ([in] double dBinning)

The "CorrelatorBinning" property specifies the time for which the pulses are accumulated before they are passed to the correlator.

pdBinning	[out]	A pointer to a variable which will receive the correlator binning time in seconds.
dBinning	[in]	The correlator binning time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorMaximumTime** ([out, retval] double* pdMaximumTime)
HRESULT **put_CorrelatorMaximumTime** ([in] double dMaximumTime)

The "CorrelatorMaximumTime" property specifies the maximum correlation time which is considered for the correlation results. In addition, the maximum correlation time is limited by the measurement time.

pdMaximumTime	[out]	A pointer to a variable which will receive the maximum correlation time in seconds.
dMaximumTime	[in]	The maximum correlation time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CorrelatorTauChannels** ([out, retval] long* plBinning)
HRESULT **put_CorrelatorTauChannels** ([in] long lBinning)

The "CorrelatorTauChannels" property specifies the number of channels that are generated per correlator tau-stage. The property determines the the density of the correlator channels.

Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticCountRateBinnig**
([out, retval] VARIANT_BOOL* pbAutomatic)
HRESULT **put_AutomaticCountRateBinnig**
([in] VARIANT_BOOL bAutomatic)

The "AutomaticCountRateBinnig" property specifies if the count rate binning time for display in

the count rate diagram is automatically ajusted by the online processing component during the acquisition. If not, the "CountRateBinnig" property is used instead.

pbAutomatic	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomatic	[in]	The value 0 indicates that the "CountRateBinnig" property is used. A different value indicates that that the binning time is determined automatically.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_CountRateBinnig** ([out, retval] double* pdBinningTime)
HRESULT **put_CountRateBinnig** ([in] double dBinningTime)

The "CountRateBinnig" property specifies the time where count rate data are averaged for display in the count rate diagram when the automatic count rate binning is switched off (see "AutomaticCountRateBinnig").

pdBinningTime	[out]	A pointer to a variable which will receive the binning time in seconds.
dBinningTime	[in]	The binning time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_AutomaticPhotonCountHistogramBinnig**
([out, retval] VARIANT_BOOL* pbAutomatic)
HRESULT **put_AutomaticPhotonCountHistogramBinnig**
([in] VARIANT_BOOL bAutomatic)

The "AutomaticPhotonCountHistogramBinnig" property specifies if the photon count histogram binning time is automatically determined by the online processing component during the acquisition. If not, the "PhotonCountHistogramBinnig" property is used instead.

pbAutomatic	[out]	A pointer to a variable which will receive the current state of the flag.
bAutomatic	[in]	The value 0 indicates that the "PhotonCountHistogramBinnig" property is used. A different value indicates that that the binning time is determined automatically.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_PhotonCountHistogramBinnig** ([out, retval] double* pdBinningTime)
HRESULT **put_PhotonCountHistogramBinnig** ([in] double dBinningTime)

The "PhotonCountHistogramBinnig" property specifies the time where count rate data are accumulated for the generation of the photon count histogram when the automatic photon count histogram binning is switched off (see "AutomaticPhotonCountHistogramBinnig").

pdBinningTime	[out]	A pointer to a variable which will receive the binning time in seconds.
dBinningTime	[in]	The binning time in seconds.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_IlluminationPeriodClocks** ([out, retval] long* plClocks)
HRESULT **put_IlluminationPeriodClocks** ([in] long lClocks)

The "IlluminationPeriodClocks" property specifies the number of detector clocks for the illumination period in fast channel switching mode.

plClocks	[out]	A pointer to a variable which will receive the currently used number of detector clocks.
lClocks	[in]	The number of detector clocks or 0 if no channel illumination switching is performed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_IlluminationPauseClocks** ([out, retval] long* plClocks)
HRESULT **put_IlluminationPauseClocks** ([in] long lClocks)

The "IlluminationPauseClocks" property specifies the number of detector clocks for the pause between the illumination in fast channel switching mode.

plClocks	[out]	A pointer to a variable which will receive the currently used number of detector clocks.
lClocks	[in]	The number of detector clocks or 0 if no channel illumination switching is performed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DestinationArraySize** ([out, retval] long* plArraySize)
HRESULT **put_DestinationArraySize** ([in] long lArraySize)

The "DestinationArraySize" method specifies the number of array elements in the of destination data sets for the correlation. Note that there is no need to use "put_DestinationArraySize" to enlarge the list. The list is enlarged automatically when "put_Destination" or "put_DestinationFlags" is called.

plArraySize	[out]	A pointer to a variable which will receive the current number of elements in the list.
lArraySize	[in]	The new number of list elements.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error

information.

HRESULT **get_Destination** ([in] long IIndex ,
[out, retval] IUnknown** ppDestination)
HRESULT **put_Destination** ([in] long IIndex ,
[in] IUnknown* pDestination)

The "Destination" property specifies the on element in the list of the destination data sets for the correlation. Note that there is no need to use "put_DestinationArraySize" to enlarge the list. The list is enlarged automatically when "put_Destination" is called.

IIndex	[in]	The zero based destination data set list index.
ppDestination	[out]	A pointer to a variable which will receive the interface pointer of the currently used data set for the list entry.
pDestination	[in]	A pointer to one interface of the destination data set object. The object must have an "IAimFcsDataSet" interface. The list entry is ignored otherwise.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DestinationFlags** ([in] long IIndex ,
[out, retval] long* plFlags)
HRESULT **put_DestinationFlags** ([in] long IIndex ,
[in] long lFlags)

The "DestinationFlags" property specifies which of the resulst generated by the correlator is written to a destination data set object. Note that there is no need to use "put_DestinationArraySize" to enlarge the list. The list is enlarged automatically when "put_DestinationFlags" is called.

IIndex	[in]	The zero based destination data set list index.
plFlags	[out]	A pointer to a variable which will receive the currently used flags.
lFlags	[in]	A conbination of the flags which indicate the correltor result types: eAimFcsProcessingDestinationCountRate - count rate eAimFcsProcessingDestinationCorrelation - correlation eAimFcsProcessingDestinationPhotonCountHistogram - photon count histogram eAimFcsProcessingDestinationPulseDistanceHistogram - pulse distance histogram
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Start** ( )

The "Start" method prepares the correlator for the processing of raw data. The internal calculation objects are set up based on the properties of the object.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Stop** ( )

The "Stop" method should be called when there are no more raw data to process. The "Stop" method will write the final result data to the result data sets.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsProcessingCorrelator

The "IAimFcsProcessingCorrelator" interface provides access to the calaculation for correlation, count rate, photon count histogram and pulse distance histogram form raw data files of multiple data sets.

Methods

HRESULT **get_List** ([out, retval] IUnknown** ppList)
HRESULT **put_List** ([in] IUnknown* pList)

The "List" property specifies the intefrcae pointer for the data set list object for which raw data files are correlated.

ppList	[out]	A pointer to a variable which will receive the interface pointer for the currenlty used data set list object.
pList	[in]	A pointer to one interface of the data set list object. The object must have an "IAimFcsDataList" interface.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_Type** ([out, retval] enumAimFcsProcessingCorrelateListType* peType)
HRESULT **put_Type** ( )

HRESULT **get_DestinationFlags** ([out, retval] long* plFlags)
HRESULT **put_DestinationFlags** ([in] long lFlags)

The "DestinationFlags" property specifies which of the results are calculated and written to the data sets.

plFlags	[out]	A pointer to a variable which will receive the currently used flags for the correlator results.
lFlags	[in]	A combination of flags which specify the type of correlator results which are calculated: eAimFcsProcessingDestinationCountRate - count rate eAimFcsProcessingDestinationCorrelation - correlation eAimFcsProcessingDestinationPhotonCountHistogram - photon count histogram eAimFcsProcessingDestinationPulseDistanceHistogram - pulse distance histogram
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method

for the component can be used to obtain detailed error information.

HRESULT **get_Fit** ([out, retval] VARIANT_BOOL* pbFit)
HRESULT **put_Fit** ([in] VARIANT_BOOL bFit)

The "Fit" property specifies if the fit is performed after the correlations for the affected data sets.

pbFit	[out]	A pointer to a variable which will receive the current state of the flag.
bFit	[in]	The value 0 indicates that the fit is not performed. A different value indicates that the fit is performed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **CanCorrelate** ([out,retval] VARIANT_BOOL* pbCanCorrelate)

The "CanCorrelate" method checks if there is at least one data set meeting the criterium of the "Type" proeprties in the data set list for which raw data files exist and the correlation can be performed.

pbCanCorrelate	[out]	A pointer to a variable which will receive "VARIANT_TRUE" if the correlation can be performed and "VARIANT_FALSE" if there are no raw data or the list or list selection is empty.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Run** ( )

The "Run" method performes correlation and optionally fit for all data sets according the object properties.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

8.15. Component AimFcsDisplay

IAimFcsDisplayTable

The "IAimFcsDisplayTable" interface is the interface of the object for display of a table and check boxes beneath the table rows in a window. The table contents are filled form selected data set properties and fit results. The check boxes are used to enable or disbale individual data sets for the calculation of averaged/concatenated data sets.

Methods

HRESULT **get_FcsData** (IUnknown** ppFcsData)
HRESULT **put_FcsData** (IUnknown* pFcsData)

The "FcsData" property specified the interface pointer for the FCS data set list to display in the table window.

ppFcsData	[out]	A pointer to a variable which will receive the pointer to the currently used FCS data set list object.
pFcsData	[in]	A pointer to one interface of the FCS data set list object for which the data should be displayed in the table.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DisplayStandardDeviation** (VARIANT_BOOL* pbStandardDeviation)
HRESULT **put_DisplayStandardDeviation** (VARIANT_BOOL bStandardDeviation)

The "DisplayStandardDeviation" property specified if the standard deviation is appended to the fist result table entries.

pbStandardDeviation	[out]	A pointer to a variable which will receive the current state of the flag.
bStandardDeviation	[in]	The value 0 indicates that the standard deviation should not be displayed. A different value indicates that the standard deviation should be displayed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SettingsPath** ([out, retval] BSTR* ppPath)
HRESULT **put_SettingsPath** ([in] BSTR pPath)

The "SettingsPath" property specified the registry path where the user settings for the display window should be stored and loaded.

ppPath	[out]	A pointer to a variable which will receive the character string with the currently used registry path.
pPath	[in]	A pointer to the character string with the new registry path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Create** ([in] aim_handle hParentWindowHandle)

The "Create" method creates the image display window. The image display window can either be a child window or a popup window. The "hParentWindowHandle" parameter is used to distinguish the two cases. "Create" must be called from a thread with a message loop.

hParentWindowHandle	[in]	Handle of the parent window. This parameter can be "NULL" if a popup window should be created. Otherwise a child window is created.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetWindowHandle** ([out,retval] aim_handle* pHandle)

The "GetWindowHandle" can be used to obtain the window handle of the image display window. The window handle can be useful to obtain further informations about the window like the current size and postion of the window.pHandle - pointer to a variable to receive the window handle. "NULL" is written to the variable if there is no valid image display window.

Returns S_OK - success E_INVALIDARG - "NULL" pointer has been passed.

HRESULT **Move** ([in] long lLeft ,
[in] long lTop ,
[in] long lWidth ,
[in] long lHeight)

"MoveWindow" canges the position and dimensions of the image display window.lLeft lTop lWidth

lHeight	[in]	new size and position of the image display window. For a popup window (the "lParentWindow" parameter in the call "Create" was "NULL") the position must be specified relative to the upper-left corner of the screen. For a child window the position must be specified relative to the upper-left corner of the client area of the parent window ("lParentWindow" parameter in "Create").
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Show** ([in] VARIANT_BOOL bShow)

The "Show" canges the visability of the image display window.

bShow	[in]	if this parameter is "0" the "Show" method will hide the window otherwise a hidden window will become visible.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **CopyToClipboard** ( )

The "CopyToClipboard" method copies the contents of the table to the clip board in text format.

Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **WriteToFile** ( )

The "WriteToFile" method opens an file save dialog and writes the table contents to the user specified text file if the user selects the Ok button in the dialog.

Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetText** ([in] enumAimFcsDisplayTableTextRows eRows ,
[out] BSTR* ppText)

The "GetText" method generates a text block form table heading and cells where the table column text is deliminated by tab characters and the table rows are deliminated by return and new-line characters. If the table heading or cells contain return or new-line characters these characters are replaced by a backslash character followed by a "r" or "n" character.

eRows	[in]	An identifier for the table rows to consider: eAimFcsDisplayTableTextRowsAll - heading and all table rows eAimFcsDisplayTableTextRowsCurrent - heading and the current row only eAimFcsDisplayTableTextRowsSelected - heading and the selected rows. If no row is selected the current row is considered.
ppText	[out]	A pointer to a variable which will receive the character string with the table contents.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PropertiesDialog** ( )

The "PropertiesDialog" method calls a modla dialog for the selection of table columns and table row sort order.

Returns	"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayCountRateDiagram

The "IAimFcsDisplayTable" interface is the interface of the object for display of the count rate data set form selected data sets in a diagram.

Methods

HRESULT **get_EditCutRegions** ([out, retval] VARIANT_BOOL* pbEditCutRegions)
HRESULT **put_EditCutRegions** ([in] VARIANT_BOOL bEditCutRegions)

The "EditCutRegions" property specified if diagram markers can be used specify the regions where count rate data are excluded form the correlation and histogram calculations.

pbEditCutRegions	[out]	A pointer to a variable which will receive the current state of the flag.
bEditCutRegions	[in]	The value 0 indicates that markers for editig of the cut regions are disabled. A different value indicates that markers are enabled.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ShowCutRegions** ([out, retval] VARIANT_BOOL* pbShowCutRegions)
HRESULT **put_ShowCutRegions** ([in] VARIANT_BOOL bShowCutRegions)

The "ShowCutRegions" property specified if the regions where count rate data are excluded form the correlation and histogram calculations are displayed in the diagram.

pbShowCutRegions	[out]	A pointer to a variable which will receive the current state of the flag.
bShowCutRegions	[in]	The value 0 indicates that the regions are not displayed. A different value indicates that regions are displayed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayCorrelationDiagram

The "IAimFcsDisplayTable" interface is the interface of the object for display of the correlation data set form selected data sets in a diagram.

IAimFcsDisplayPhotonCountHistogram

The "IAimFcsDisplayTable" interface is the interface of the object for display of the photon count histogram data set form selected data sets in a diagram.

IAimFcsDisplayPulseDistanceHistogram

The "IAimFcsDisplayTable" interface is the interface of the object for display of the pulse distance histogram data set form selected data sets in a diagram.

IAimFcsDisplayFitDiagram

The "IAimFcsDisplayTable" interface is the interface of the object for display of the fit result and deviation data set form selected data sets in a diagram.

Methods

HRESULT **get_ShowMeasuredData**
([out, retval] VARIANT_BOOL* pbShowMeasuredData)
HRESULT **put_ShowMeasuredData**
([in] VARIANT_BOOL bShowMeasuredData)

The "ShowMeasuredData" property specifies if the graph for the measured dataresult is displayed in the diagram.

pbShowMeasuredData	[out]	A pointer to a variable which will receive the current state of the flag.
bShowMeasuredData	[in]	The value 0 indicates that the graph for the measured data is not displayed. A different value indicates that the graph for the measured data is displayed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ShowFitResult** ([out, retval] VARIANT_BOOL* pbShowFitResult)
HRESULT **put_ShowFitResult** ([in] VARIANT_BOOL bShowFitResult)

The "ShowFitResult" property specifies if the graph for the theorethical data from the fit results is displayed in the diagram.

pbShowFitResult	[out]	A pointer to a variable which will receive the current state of the flag.
bShowFitResult	[in]	The value 0 indicates that the graph for the fit result is not displayed. A different value indicates that the graph for the fit result is displayed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ShowDeviation** ([out, retval] VARIANT_BOOL* pbShowDeviation)
HRESULT **put_ShowDeviation** ([in] VARIANT_BOOL bShowDeviation)

The "ShowDeviation" property specifies if the graph for the difference the measured values from

the theorethical data for the fit results is displayed in the diagram.

pbShowDeviation	[out]	A pointer to a variable which will receive the current state of the flag.
bShowDeviation	[in]	The value 0 indicates that the graph for the deviation is not displayed. A different value indicates that the graph for the deviation is displayed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_FitRangeMarkers** ([out, retval] VARIANT_BOOL* pbFitRangeMarkers)
HRESULT **put_FitRangeMarkers** ([in] VARIANT_BOOL bFitRangeMarkers)

The "FitRangeMarkers" property specifies if the markers for the fit range are displayed and active.

pbFitRangeMarkers	[out]	A pointer to a variable which will receive the current state of the flag.
bFitRangeMarkers	[in]	The value 0 indicates that the markers for the fit range are not displayed. A different value indicates that the markers are displayed and active.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_DeviationScaleRange** ([out, retval] double* pdDeviationScaleRange)
HRESULT **put_DeviationScaleRange** ([in] double dDeviationScaleRange)

The "DeviationScaleRange" property specifies minimum ordinate scale range for the fit deviation diagram. The scale starts at least at - "DeviationScaleRange" and ends at least at + "DeviationScaleRange".

pdDeviationScaleRange	[out]	A pointer to a variable which will receive the currently used minimum ordinate scale range for the fit deviation diagram.
dDeviationScaleRange	[in]	The new minimum ordinate scale range for the fit deviation diagram.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayFitParameters

The "IAimFcsDisplayFitParameters" interface is the interface of the object for a window for editing of fit parameters and the start of the fit operation. The fit parameters are always editied for the current data set. The fit can be started for the current and linked data sets or for all data sets. Undo and redo functions are supported.m_Lock - The object which is used to synchronize the access to properties form different threads.

Methods

HRESULT **get_FcsData** (IUnknown** ppFcsData)
HRESULT **put_FcsData** (IUnknown* pFcsData)

The "FcsData" property specified the interface pointer for the FCS data set list to display in the table window.

ppFcsData	[out]	A pointer to a variable which will receive the pointer to the currently used FCS data set list object.
pFcsData	[in]	A pointer to one interface of the FCS data set list object for which the data should be displayed in the table.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_ShowLimits** ([out, retval] VARIANT_BOOL* pbShowLimits)
HRESULT **put_ShowLimits** ([in] VARIANT_BOOL bShowLimits)

The "ShowLimits" method specifies if the minimum and maximum limits for the possible range of the fit result is displayed in the table.

pbShowLimits	[out]	A pointer to a variable which will receive the current state of the flag.
bShowLimits	[in]	The value 0 indicates that the upper and lower limits are displayed in the data grid. All other values indicate that the limits are not displayed.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SettingsPath** ([out, retval] BSTR* ppPath)
HRESULT **put_SettingsPath** ([in] BSTR pPath)

The "SettingsPath" property specified the registry path where the user settings for the display

window should be stored and loaded.

ppPath	[out]	A pointer to a variable which will receive the character string with the currently used registry path.
pPath	[in]	A pointer to the character string with the new registry path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_FcsController** (IUnknown** ppFcsController)
HRESULT **put_FcsController** (IUnknown* pFcsController)

The "FcsController" property specified the interface pointer for the FCS hardware controller which is used for the "Write to method" function for fit parameters.

ppFcsController	[out]	A pointer to a variable which will receive the currently used pointer of the FCS hardware controller.
pFcsController	[in]	A pointer to one interface of the FCS hardware controller.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Create** ([in] aim_handle hParentWindowHandle)

The "Create" method creates the image display window. The image display window can either be a child window or a popup window. The "hParentWindowHandle" parameter is used to distinguish the two cases. "Create" must be called from a thread with a message loop.

hParentWindowHandle	[in]	Handle of the parent window. This parameter can be "NULL" if a popup window should be created. Otherwise a child window is created.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetWindowHandle** ([out,retval] aim_handle* pHandle)

The "GetWindowHandle" can be used to obtain the window handle of the image display window. The window handle can be useful to obtain further informations about the window like the current size and postion of the window.pHandle - pointer to a variable to receive the window handle. "NULL" is written to the variable if there is no valid image display window.

Returns S_OK - success E_INVALIDARG - "NULL" pointer has been passed.

HRESULT **Move** ([in] long lLeft ,
[in] long lTop ,

[in] long lWidth ,
[in] long lHeight)

"MoveWindow" canges the position and dimensions of the image display window.lLeft lTop lWidth

lHeight	[in]	new size and position of the image display window. For a popup window (the "lParentWindow" parameter in the call "Create" was "NULL") the position must be specified relative to the upper-left corner of the screen. For a child window the position must be specified relative to the upper-left corner of the client area of the parent window ("lParentWindow" parameter in "Create").
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Show** ([in] VARIANT_BOOL bShow)

The "Show" canges the visability of the image display window.

bShow	[in]	if this parameter is "0" the "Show" method will hide the window otherwise a hidden window will become visible.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayTable

The "IAimFcsDisplayTable" interface is the interface of the window which displays a coincidence plot form the count rate data of the tow channels with the same coordinate indexes (except channel index) of the current data set in a data set list. The coincidence plot with scaling for the count rate and a sale bar for the frequency.

Methods

HRESULT **get_FcsData** (IUnknown** ppFcsData)
HRESULT **put_FcsData** (IUnknown* pFcsData)

The "FcsData" property specified the interface pointer for the FCS data set list to display in the table window.

ppFcsData	[out]	A pointer to a variable which will receive the pointer to the currently used FCS data set list object.
pFcsData	[in]	A pointer to one interface of the FCS data set list object for which the data should be displayed in the table.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Create** ([in] aim_handle hParentWindowHandle)

The "Create" method creates the display window. The display window can either be a child window or a popup window. The "hParentWindowHandle" parameter is used to distinguish the two cases. "Create" must be called from a thread with a message loop.

hParentWindowHandle	[in]	Handle of the parent window. This parameter can be "NULL" if a popup window should be created. Otherwise a child window is created.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetWindowHandle** ([out,retval] aim_handle* pHandle)

The "GetWindowHandle" can be used to obtain the window handle of the display window. The window handle can be useful to obtain further informations about the window like the current size and postion of the window.pHandle - pointer to a variable to receive the window handle. "NULL" is

written to the variable if there is no valid display window.

Returns S_OK - success E_INVALIDARG - "NULL" pointer has been passed.

HRESULT **Move** ([in] long lLeft ,
[in] long lTop ,
[in] long lWidth ,
[in] long lHeight)

"MoveWindow" canges the position and dimensions of the display window.lLeft lTop lWidth

lHeight [in] new size and position of the display window. For a popup window (the "lParentWindow" parameter in the call "Create" was "NULL") the position must be specified relative to the upper-left corner of the screen. For a child window the position must be specified relative to the upper-left corner of the client area of the parent window ("lParentWindow" parameter in "Create").

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Show** ([in] VARIANT_BOOL bShow)

The "Show" canges the visability of the display window.

bShow [in] if this parameter is "0" the "Show" method will hide the window otherwise a hidden window will become visible.

Returns "S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayCountrateDisplay

The "IAimFcsDisplayCountrateDisplay" interface is the interface of the modeless dialog window for online display of of count rate, correlation or counts per molecule. Up to three numeric displays are shown depending on the current measurement parameters in the acuqisition controller. Two displays are used for the two detectors. The third display is used for cross correlation. The display type, unit and averaging vcan be changed with buttons. The displays have an intesity bar for the displayed values.

Methods

HRESULT **get_FcsController** (IUnknown** ppFcsController)
HRESULT **put_FcsController** (IUnknown* pFcsController)

The "FcsController" property specifies the interface pointer for the FCS device controller whith the current acquisition parameters and online calculations for countrate, correlation and counts per molecule.

ppFcsController	[out]	A pointer to a variable which will receive the pointer to the currently used FCS hardware controller object.
pFcsController	[in]	A pointer to one interface of the FCS hardware controller object.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **get_SettingsPath** ([out, retval] BSTR* ppPath)
HRESULT **put_SettingsPath** ([in] BSTR pPath)

The "SettingsPath" property specified the registry path where the user settings for the display window should be stored and loaded.

ppPath	[out]	A pointer to a variable which will receive the character string with the currently used registry path.
pPath	[in]	A pointer to the character string with the new registry path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Create** ([in] aim_handle hParentWindowHandle)

The "Create" method creates the display window. The display window can either be a child

window or a popup window. The "hParentWindowHandle" parameter is used to distinguish the two cases. "Create" must be called from a thread with a message loop.

hParentWindowHandle	[in]	Handle of the parent window. This parameter can be "NULL" if a popup window should be created. Otherwise a child window is created.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **GetWindowHandle** ([out,retval] aim_handle* pHandle)

The "GetWindowHandle" can be used to obtain the window handle of the display window. The window handle can be useful to obtain further informations about the window like the current size and postion of the window.pHandle - pointer to a variable to receive the window handle. "NULL" is written to the variable if there is no valid display window.

Returns S_OK - success E_INVALIDARG - "NULL" pointer has been passed.

HRESULT **Move** ([in] long lLeft ,
[in] long lTop ,
[in] long lWidth ,
[in] long lHeight)

"MoveWindow" canges the position and dimensions of the display window.lLeft lTop lWidth

lHeight	[in]	new size and position of the display window. For a popup window (the "lParentWindow" parameter in the call "Create" was "NULL") the position must be specified relative to the upper-left corner of the screen. For a child window the position must be specified relative to the upper-left corner of the client area of the parent window ("lParentWindow" parameter in "Create").
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Show** ([in] VARIANT_BOOL bShow)

The "Show" canges the visability of the display window.

bShow	[in]	if this parameter is "0" the "Show" method will hide the window otherwise a hidden window will become visible.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayPinholeAdjustmentDialog

The "IAimFcsDisplayPinholeAdjustmentDialog" interface is the interface of the dialog for display of the intensities of an maximum intensity search for one of the pinhole coordinates x, y or z. The "PinholeAdjustment" method cretates a modal dialog and starts the acquisition of the intensities. The "PinholeAdjustment" method waits for the user to close the dialog.

Methods

HRESULT **get_SettingsPath** ([out, retval] BSTR* ppPath)
HRESULT **put_SettingsPath** ([in] BSTR pPath)

The "SettingsPath" property specifies the registry path where the user settings for the pinhole adjustment dialog should be stored and loaded.

ppPath	[out]	A pointer to a variable which will receive the character string with the currently used registry path.
pPath	[in]	A pointer to the character string with the new registry path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **PinholeAdjustment**

(	IUnknown*	pFcsController	,
	aim_handle	hParentWindowHandle	,
	long	lDetector	,
	enumAimFcsDisplayPinholeCoordinate	eCoordinate	,
	VARIANT_BOOL	bFineAdjustment	)

The "PinholeAdjustment" method creates a modal dialog and performs a maximum intensity search for the specified pinhole coordinate. The method returns after the user has closed the dialog box.

pFcsController	[in]	The pointer to the interface of the FCS hardware control object which is used for the data acquisition.
hParentWindowHandle	[in]	The handle of the parent window for the dialog. The dialog is cenered over the parent window.
lDetector	[in]	The zero based index of the detector which should be used. For some devices the detector index determines the pinhole which is adjusted.
eCoordinate	[in]	The identifier of the pinhole coordinate: eAimFcsDisplayPinholeCoordinateX - x - coordinate eAimFcsDisplayPinholeCoordinateY - y - coordinate eAimFcsDisplayPinholeCoordinateZ - z - coordinate

bFineAdjustment	[in]	The value 0 indicates that the whole drive range is checked for maximum intensity. A different value indicates that the meighbourhood of the current position is checked.

IAimFcsDisplayScanWindow

The "IAimFcsDisplayScanWindow" interface is the interface of the dialog for display of the intensities of an one or two dimensional scan in x-, y- and z-direction with stage or scanners and focus drive. The "Scan" method cretates a modal dialog and starts the acquisition of the intensities. The "Scan" method waits for the user to close the dialog.

Methods

HRESULT **get_SettingsPath** ([out, retval] BSTR* ppPath)
HRESULT **put_SettingsPath** ([in] BSTR pPath)

The "SettingsPath" property specifies the registry path where the user settings for the scan display window should be stored and loaded.

ppPath	[out]	A pointer to a variable which will receive the character string with the currently used registry path.
pPath	[in]	A pointer to the character string with the new registry path.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

HRESULT **Scan** ([in] IUnknown* pFcsController ,
[in] aim_handle hParentWindowHandle ,
[in] double dStartX ,
[in] double dStartY ,
[in] double dStartZ ,
[in] double dEndX ,
[in] double dEndY ,
[in] double dEndZ ,
[in] long lPositionsX ,
[in] long lPositionsY ,
[in] long lPositionsZ)

The "Scan" method creates a modal dialog and performs an one or two dimensional scan. The method returns after the user has closed the dialog box.

pFcsController	[in]	The pointer to the interface of the FCS hardware control object which is used for the data acquisition.
hParentWindowHandle	[in]	The handle of the parent window for the dialog. The dialog is cenered over the parent window.
dStartZ	[in]	The stage and focus positions for the acquisition of the first count rate result.
dEndZ	[in]	The stage and focus positions for the acquisition of the last

		count rate result.
lPositionsZ	[in]	The number of pixles to acquire in the corresponding stage and focus directions.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayReCorrelateDialog

The "IAimFcsDisplayReCorrelateDialog" interface is the interface of the dialog for selection selection of the correlator properties. If the user presses the Ok-button in the dialog the correlation form raw data files is started.

Methods

HRESULT **ReCorrelate** ([in] aim_handle hParentWindowHandle ,
[in] IUnknown* pDataSet)

The "ReCorrelate" method creates a modal dialog for the selection of the parmeters of the correlator. If the user presses the Ok button the correlation is started for all selected data sets.

hParentWindowHandle	[in]	The handle of the parent window for the dialog.
pDataSet	[in]	A pointer to the object with the result data sets. The object must have an "IAimFcsDataSet" interface.
Returns		"S_OK" if successful or an error code. In the latter case the "LastError" method for the component can be used to obtain detailed error information.

IAimFcsDisplayDiagram

The "IAimFcsDisplayDiagram" common interface for the FCS diagram display objects.