PROGRAMMER'S REFERENCE MANUAL

for the

PAKON F235 SCANNERS

TLA Version 0.0.30.2

15 June 2004

Disclaimer

Trademarks

Pakon Inc
5950 Clearwater Drive
Suite 100
Minnetonka, MN 55343
(952) 936-9500
www.pakon.com

Table of Contents

1 Introduction

The Pakon F235*plus* and F235C Scanners are input scanners that convert images from developed film into digital images. They can scan APS or 35mm color negative, color reversal, or black & white film. The filmstrips can be as short as two frames and as long as forty frames. The F235C has a built-in APS cartridge loader and MOF reader. The F235*plus* can scan APS film, but it has no cartridge loader, the film must be manually fed in. It also has no MOF reader.

The scanner will be connected to a host computer via a USB 2.0 cable, and software on the host will need to be developed that communicates with and controls the scanner. A software development kit (SDK) is supplied with the scanner that allows programmers to do this. This SDK consists of a DLL that utilizes a COM interface.

Through the COM interface, the client software can operate the film scanner, including setting parameters for framing and cropping, scanning, performing color corrections, and saving pictures to disk or memory. The scanner will utilize buffer space on the host computer in order to provide for fast scans, and to allow complete control over the final product, digital images.

The main purpose of this document is to provide a reference for all the functions available in this COM interface. See the *Programmer's User Guide for the Pakon F235 Scanner* for more explanations on how the scanner works and sample scanning scenarios.

2 Template

Each available function or method is listed using the following template:

Methodname

<Description of the method>

```
HRESULT Methodname(type1 param1,
                   type2 param2);
```

Parameters

param1
[in] <Description…>

param2
[out] <Description…>

Return Value

<The possible return values>

Additional Error Codes

<A list of the possible error codes returned by the GetAndClearLastError method>

Remarks

<Additional remarks about how to use the method>

Example

<An example call to this method>

Callbacks

<A table of the possible callback messages>

See Also

<A list of other related methods>

Note that not all listings will include an example, and only long operations will include a table of callback messages.

3 Function Reference

This section describes the methods available through the F235 COM Server. It is organized into sections that correspond to the interfaces of the COM Server. There are five interfaces: Long Ops CB, CallBackClient, TLA Main, Scan Pictures, and Save Pictures. Within each section, the available functions are listed alphabetically.

3.1 Long Ops CB

This interface establishes the callback mechanism with the server. The callback is used mainly in long operations, but also to report some errors.

3.1.1 CBAdvise

The ILongOpsCB::CBAdvise method creates a connection between the callback client interface and the COM Server (TLA).

```
HRESULT CBAdvise(ICallBackClient *pICallBackClient,
                 long *plCookie);
```

Parameters

pICallBackClient

[in] A pointer to the callback interface created by the client.

plCookie

[out] A pointer to an identifier of the callback connection. This identifier can be used later to break the connection using CBUnadvise.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InitializeScannerAlreadyCalled
EC_CBAdviseAlreadyCalled EC_StartUpError EC_InvalidPtrToClientCallback

Remarks

Call this function after COM is initialized and the client's callback interface is defined, but before the scanner is initialized. Calling CBAdvise will cause an AddRef on the pICallBackClient, so it is necessary to call Release on the pICallBackClient after calling this function. When the client is ready to shutdown, it must call CBUnadvise in order to remove the callback connection with TLA

Example

```
// Create a callback object
CCallBackClient *pICallBackClient = new CCallBackClient();
// Set up the callback connection
hr = m_pILongOpsCB->CBAdvise(pICallBackClient, &m_lCookie);
if(FAILED(hr))
{
   ::AnalyzeComError(hr, IID_ILongOpsCB, m_pILongOpsCB,
                     NULL, FALSE);
   pICallBackClient->Release();
   return FALSE;
}
// Done with our ref count.  Server did an AddRef
pICallBackClient->Release();
```

See Also

CBUnadvise

3.1.2 CBUnadvise

The ILongOpsCB::CBUnadvise method breaks the connection between the COM server (TLA) and the client callback interface.

HRESULT CBUnadvise(long *lCookie*);

Parameters

lCookie
[in] The identifier of the callback connection. This was established when CBAdvise was called.

Return Value

If successful, S_OK is returned, otherwise an error code of E_NOINTERFACE will be returned.

Additional Error Codes
None

Remarks

Call this function when the client is ready to shutdown, but before the callback interface is released.

Example

```
HRESULT hr = m_pILongOpsCB->CBUnadvise(m_lCookie);
if(FAILED(hr))
   ::AnalyzeComError(hr, IID_ILongOpsCB, m_pILongOpsCB,
                     NULL, FALSE);

// Release the server object
m_pILongOpsCB->Release();
```

See Also
CBAdvise

3.2 CallBackClient

This interface is defined by the client and allows the server to send progress and error messages back to the client.

3.2.1 Awake

The COM Server (TLA) will call the ICallBackClient::Awake method to inform the client of an error or to report progress of a long operation. It is also used to report unsolicited hardware errors, such as a lamp burnout. The client must create this method.

```
HRESULT Awake(long lOperation,
              long lStatus);
```

Parameters

lOperation

[in] An indicator of the operation being reported on. Possible values can be found in the WORKER_THREAD_OPERATION_000 enumeration.

lStatus

[in] An indicator of the status or error. See the WORKER_THREAD_PROGRESS_000 and the HARDWARE_CB_000 enumerations for possible values.

Return Value

If successful, S_OK should be returned, otherwise some meaningful error code from the ERROR_CODES_xxx enumerations in section 5 Error Codes should be returned.

Additional Error Codes

None

Remarks

The Awake method can be defined any way the client sees fit. The restriction that the method return quickly so as not to tie up the COM Server has been removed.

Example

This example from TLAClientDemo simply sends a message to the main window:

```
STDMETHODIMP CCallBackClient::Awake(long lOperation,
   long lStatus)
{
   AfxGetMainWnd()->PostMessage(WM_COM_CALLBACK,
                                (ULONG)lOperation,
                                lStatus);
   return S_OK;
}
```

3.3 TLA Main

This interface provides the main functions of TLA, including initializing the scanner and handling errors. The operation InitializeScanner takes a while to complete and so gets its own processing thread. It is called a "long operation" and is marked as such in the function description. When this function is called, it starts its own processing thread, then returns immediately.

The following is a list of all the TLA Main functions in alphabetical order.

3.3.1 GetAndClearLastError

The ITLAMain::GetAndClearLastError method gets the last error, gives a call stack trace for analysis, and clears the error.

```
HRESULT GetAndClearLastError(int iShort_IID,
                             BSTR *pbstrError,
                             BSTR *pbstrErrorNumbers,
                             int *piError);
```

Parameters

iShort_IID

[in] Specifies in which interface the error occurred. See the INT_IID_000 enumeration for valid values.

pbstrError

[out] Text denoting the call stack trace of the error; this includes class name, function name, error name and other information.

pbstrErrorNumbers

[out] Text denoting the call stack trace of the error; this includes class number, function number, error number and other information. See the CLASS_NAMES_000 and FUNCTION_NAMES_000 enumerations in the TLA.idl, and the ERROR_CODES_xxx enumerations in section 5 Error Codes to relate the numbers to meaningful names.

piError

[out] The error number. See the ERROR_CODES_xxx enumerations in section 5 Error Codes for possible values.

Return Value

S_OK is returned.

Remarks

This method should be called when the server reports an error in either a server function call (HRESULT <> S_OK) or a progress callback (an Awake call with operation=WTO_xxxError).

As an aid in troubleshooting, this function gives a call stack trace of what the error was and which class and function started the process that resulted in the error. This information is also entered into a text error log that is saved to disk. The filename depends on the interface, but will be either PakonErrorLogMain.txt or PakonErrorLogScan.txt or PakonErrorLogSave.txt. These files can be found in the "C:\Program Files\Pakon\TLA COM Server" folder.

3.3.2 GetInitializeWarnings

The ITLAMain::GetInitializeWarnings method will report any warnings during scanner initialization.

```
HRESULT GetInitializeWarnings(int *piInitializeWarnings);
```

Parameters

piInitializeWarnings

[out] The bitwise sum of warning codes. For a list of possible warning codes, see the INITIALIZE_WARNINGS_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_ScannerNotInitialized

Remarks

This method should be called after InitializeScanner is called.

Example

```
int iInitializeWarnings;
HRESULT hr =
       m_pITLAMain->GetInitializeWarnings(&iInitializeWarnings);

if(FAILED(hr))
{
   ::AnalyzeComError(hr, IID_ISavePictures, m_pISavePictures,
                     NULL);
   return;
}
// check for errors
if (iInitializeWarnings & INITIALIZEW_EEPROM_BLANK)
{
   // EEPROM not initialized
}
else if(iInitializeWarnings & INITIALIZEW_EEPROM_CHECKSUM_BAD)
{
   // EEPROM read failed
}
```

See Also

InitializeScanner

3.3.3 InitializeScanner

The ITLAMain::InitializeScanner method initializes the scanner and TLA.

```
HRESULT InitializeScanner(int iInitializeControl,
                          int iSaveToMemoryTimeout,
                          int i_uiSaveToSharedMemorySize);
```

Parameters

iInitializeControl

[in] A logical sum of the options to be included in the initialization. Valid choices can be found in the INITIALIZE_CONTROL_000 enumeration.

iSaveToMemoryTimeout

[in] The number of milliseconds the COM Server will wait for a buffer in a call to the SaveToClientMemory function. The valid range is 1000 to INFINITE.

i_uiSaveToSharedMemorySize

[in] Reserved for future implementation.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_StartUpError EC_InvalidParameter
EC_InitializeScannerAlreadyCalled EC_FileNotFound EC_QueryInterface
EC_CoMarshalInterThreadInterfaceInStream EC_UnableToCreateWorkerThread
EC_WorkerThreadCoInitialize EC_WorkerThreadCoGetInterfaceAndReleaseStream
EC_WorkerThreadClientSignal EC_WorkerThreadStartTimeout EC_SystemInfo
EC_MissingDllFunction EC_CBAdviseNotCalled EC_PicVersion
EC_EEPromWarningBlank EC_EEPromWarningCheckSumBad
EC_FirmwareVerification EC_SelfTestFailedCcdStepper
EC_SelfTestFailedFilmDrive EC_SelfTestFailedLensStepper

Remarks

This method must be called after COM is initialized and the client callback is established, but before any scanner functions are used. This method must be called only once. Once called, the client can call GetInitializeWarnings to retrieve any warning messages. This is a long operation, a separate thread is created and this function returns immediately. Once started, the client will get a progress message through the callback interface when complete.

Only one instance of TLA can be running at any time. If two clients try to start TLA, it may succeed, but TLA will not run properly.

This method should be called such that the *iInitializeControl* parameter includes the enum value INITIALIZE_FirmwareUpdate, otherwise the scanner may run with older firmware. It is also recommended that the INITIALIZE_MotorSelfTest option be included.

Callbacks

Operation	Status	Comment
WTO_InitializeProgress	WTP_Initialize	Always, right after WT startup
WTO_FirmwareUpdateApsProgress	WTP_ProgressStart	If client requested update and update is required
WTO_FirmwareUpdateApsProgress	WTP_ProgressComplete	If client requested update and update is required
WTO_FirmwareUpdateCcdProgress	WTP_ProgressStart	If client requested update and update is required
WTO_FirmwareUpdateCcdProgress	WTP_ProgressComplete	If client requested update and update is required
WTO_FirmwareUpdateDxProgress	WTP_ProgressStart	If client requested update and update is required
WTO_FirmwareUpdateDxProgress	WTP_ProgressComplete	If client requested update and update is required
WTO_FirmwareUpdateLampProgress	WTP_ProgressStart	If client requested update and update is required
WTO_FirmwareUpdateLampProgress	WTP_ProgressComplete	If client requested update and update is required
WTO_FirmwareUpdateMotorProgress	WTP_ProgressStart	If client requested update and update is required
WTO_FirmwareUpdateMotorProgress	WTP_ProgressComplete	If client requested update and update is required
WTO_ExerciseSteppersProgress	WTP_ProgressStart	If client requested
WTO_ExerciseSteppersProgress	WTP_ProgressComplete	If client requested
WTO_InitializeProgress	WTP_ProgressStart	Always
WTO_InitializeProgress	WTP_ProgressComplete	If successful
WTO_InitializeError	WTP_ProgressComplete	If error
WTO_FirmwareUpdateError	WTP_ProgressComplete	If error downloading
WTO_ExerciseSteppersError	WTP_ProgressComplete	If error with steppers

See Also

GetInitializeWarnings GetScannerInfo000 Awake
SaveToClientMemory

3.4 Scan Pictures

The functions in this section all concern setting up scanning parameters and scanning film. Some of the operations take a while to complete and so these operations get their own processing thread. These are called "long scan operations" and are marked as such in the function description. When these functions are called, they start their own processing thread, then return immediately.

The long scan operations include:

AdvanceFilm FilmTrackTest ForceCorrections
PutFilmGuidePosition PutFilmPressureRollerPosition ScanPictures

When a long scan operation is running, the client cannot call any other function from the Scan Pictures interface with the exception of the ScanCancel function. However, the client can call any function from the Save Pictures interface.

The following is a list of all the Scan Picture functions in alphabetical order.

3.4.1 AdjustMotorSpeed

The IScanPictures::AdjustMotorSpeed method adjusts the speed of the film transport motor.

```
HRESULT AdjustMotorSpeed();
```

Parameters

none

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_ScannerNotInitialized

Remarks

Call this method if the piScanWarnings parameter of the GetPictureCountScanGroup method indicates that the motor speeds need to be adjusted. The amount of adjustment is determined from the piScanWarnings. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

GetPictureCountScanGroup

3.4.2 AdvanceFilm

The IScanPictures::AdvanceFilm method advances the film in the film transport forward or backwards at a specified speed and for a specified length of time.

```
HRESULT AdvanceFilm(int iAdvanceMilliseconds,
                    int iAdvanceSpeed);
```

Parameters

iAdvanceMilliseconds

[in] The time to advance the film transport in milliseconds. Valid values are –1 and 1 to 5 X 60000, inclusive. A value of –1 will cause the transport to run forever (until ScanCancel is called). NOTE: on some computers, the actual time may vary by 10-15%.

iAdvanceSpeed

[in] The speed at which to advance the film transport in tenths of millimeters per second. Valid values are anything except 0. Negative values transport the film backwards. The value passed in may be shortened by the maximum or minimum speed of the motor.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter EC_ScannerNotInitialized
EC_UnableToCreateWorkerThread EC_WorkerThreadClientSignal EC_TimeOut
EC_WorkerThreadStartTimeout EC_MemoryNew EC_WIN_SetEvent
EC_WIN_ResetEvent EC_WIN_DeviceIoControl EC_WIN_WaitForSingleObject
EC_WIN_FileOpen EC_WIN_FileClose EC_WIN_GetOverlappedResult
EC_DRV_InvalidPacketType EC_DRV_PacketHostError* EC_DRV_PacketBusy
EC_DRV_PacketCmdErr EC_DRV_PacketCommErr

Remarks

This is a long scan operation. This method cannot be called if another long scan operation is in progress, an error will occur. Once started, the client will get progress messages through the callback interface until complete. However, the client can stop the film advance prematurely by calling ScanCancel. WARNING: If the end of the film is already inside the scanner and the film is reversed, it is possible for the film to jam.

Callbacks

Operation	Status	Comment
WTO_AdvanceFilmProgress	WTP_Initialize	Always, right after WT startup
WTO_AdvanceFilmProgress	WTP_ProgressComplete	If successful
WTO_AdvanceFilmError	WTP_ProgressComplete	If error

See Also

ScanCancel Awake

3.4.3 ApsManualRetract

The IScanPictures::ApsManualRetract method manually retracts APS film back into its cartridge. The MOF information is not read. This method is valid only on F235C scanners.

```
HRESULT ApsManualRetract();
```

Parameters

none

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned. If the scanner is an F235, E_NOTIMPL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_WIN_FileOpen EC_WIN_GetOverlappedResult
EC_WIN_ResetEvent EC_WIN_WaitForSingleObject EC_WIN_DeviceIoControl
EC_TimeOut EC_DRV_InvalidPacketType EC_DRV_PacketBusy
EC_DRV_PacketCmdErr EC_DRV_PacketCommErr EC_InvalidParameter
EC_ApsFilmJamRetract EC_ApsPark EC_ApsParkInit

Remarks

Use this method to retract APS film that has failed to retract itself. It can be called whether the film is moving or not. The retraction will run for a set amount of time. However, the client can stop it prematurely by calling ScanCancel. This method cannot be called if a long scan operation is in progress, an error will occur.

Callbacks

Operation	Status	Comment
WTO_F235C_ManualRetractProgress	WTP_Initialize	Always, after WT startup
WTO_F235C_ManualRetractProgress	WTP_ProgressStart	Always
WTO_F235C_ManualRetractProgress	WTP_ProgressEnd	If successful
WTO_F235C_ManualRetractProgress	WTP_ProgressComplete	If successful

See Also

ScanPictures ScanCancel

3.4.4 CountRemainingMofFiles

The IScanPictures::CountRemainingMofFiles method is reserved for future implementation.

```
HRESULT CountRemainingMofFiles(BOOL bDeleteOldest,
                               int *piFilesRemaining);
```

Parameters

bDeleteOldest

piFilesRemaining

Return Value

Additional Error Codes

Remarks

See Also

3.4.5 FilmTrackTest

The IScanPictures::FilmTrackTest method performs tests and calibrations on the film track. To get the results of these tests, use FilmTrackTestResults or look at the log file. Use this test if the scanner is having trouble reading DX codes.

```
HRESULT FilmTrackTest(BOOL pbDxPotsAdjust
                      int iFilmFormat);
```

Parameters

pbDxPotsAdjust

[in] Not used. Always set to False.

iFilmFormat

[in] The format of film for which the test will take place. Valid values can be found in the FILM_FORMAT_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NotSupportedByHW EC_ScannerNotInitialized
EC_UnableToCreateWorkerThread EC_WorkerThreadClientSignal
EC_WorkerThreadStartTimeout EC_DRV_InvalidPacketType
EC_DRV_PacketBusy EC_DRV_PacketChecksumErr EC_DRV_PacketCmdErr
EC_DRV_PacketCommErr EC_DRV_PacketHostError* EC_WIN_DeviceIoControl
EC_WIN_FileOpen EC_WIN_GetOverlappedResult EC_WIN_ResetEvent
EC_StepperAlreadyMoving EC_StepperDidNotStop EC_DXNoFilmFound
EC_DXBadSwing

Remarks

Film with DX codes must be fed into the scanner for this test. The results can be retrieved with a call to FilmTrackTestResults or by looking at the log file, PakonFilmTrackTestLog.txt, found in the “C:\Program Files\Pakon\TLA COM Server” folder.

This is a long scan operation. This method cannot be called if another long scan operation is in progress, an error will occur. Once started, the client will get progress messages through the callback interface until complete. However, the client can stop the tests prematurely by calling ScanCancel.

Callbacks

Operation	Status	Comment
WTO_FilmTrackTestProgress	WTP_Initialize	Always, right after WT startup
WTO_FilmTrackTestProgress	WTP_ProgressStart	Always
WTO_FilmTrackTestProgress	WTP_ProgressComplete	If successful
WTO_FilmTrackTestError	WTP_ProgressComplete	If error

See Also

FilmTrackTestResults Awake ScanCancel

3.4.6 FilmTrackTestResults

The IScanPictures::FilmTrackTestResults method returns the results of the FilmTrackTest.

```
HRESULT FilmTrackTestResults(int *piErrors);
```

Parameters

piErrors

[out] An integer representing the results. This will be 0 if no errors occurred or a logical sum of error codes from the FILM_TRACK_TEST_ERRORS_000 enumeration. If *piErrors* is 0xFF, this indicates that the track test was aborted for some reason.

Return Value

S_OK is returned.

Additional Error Codes

None

Remarks

This method can be called anytime.

The results can also be viewed from the log file, "PakonFilmTrackTestLog.txt", found in the "C:\Program Files\Pakon\TLA COM Server" folder.

Example

```
HR = FilmTrackTestResults(piErrors);
if (piErrors & FILM_TRACK_TEST_ERRORS_ClockBottom)
{
   // Clock bottom sensor bad
}
```

See Also

FilmTrackTest

3.4.7 ForceCorrections

The IScanPictures::ForceCorrections method causes a certain type of correction to be performed. The type of correction will depend on the *iCalibrateControl* parameter. Three types are possible: gain & offset, focus, and fixed pattern.

```
HRESULT ForceCorrections(int iResolution,
                         int iFilmColor,
                         int iFilmFormat,
                         int iCalibrateControl);
```

Parameters

iResolution

[in] The scan resolution for which correction will take place. Valid values can be found in the RESOLUTION_000 enumeration.

iFilmColor

[in] The type of film for which correction will take place. Valid values can be found in the FILM_COLOR_000 enumeration.

iFilmFormat

[in] The format of film for which correction will take place. Valid values can be found in the FILM_FORMAT_000 enumeration.

iCalibrateControl

[in] Indicates the type of correction to be performed. Valid values can be found in the CALIBRATE_CONTROL_000 enumeration. Gain & offset, and fixed pattern can be done together (by summing values), but focus must be done alone.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter EC_ScannerNotInitialized
EC_UnableToCreateWorkerThread EC_WorkerThreadClientSignal
EC_WorkerThreadStartTimeout EC_EEPromCorrupted EC_StepperDidNotStop

Remarks

Gain & offset correction: This should be performed after the scanner is initialized, but before any scanning is performed. It must be called once for every possible combination of resolution, film color and film format that the operator will perform. This will setup the scanner for proper scanning for these combinations. If this correction is not done, subsequent scans will produce saturated pictures of very poor quality. There must be NO film in the scanner.

Focus correction: This will adjust the focus of the scanner. Usually, the client won't have to correct this, but this could be performed if the focus seems off. This can be done with film already in the scanner (`iCalibrateControl` =CALIBRATE_Focus) or film ready to be fed in (`iCalibrateControl` =CALIBRATE_FocusAdvanceFilm). If focus correction is being done, is must be done by itself, otherwise an error will occur.

Fixed pattern correction: This will correct for non-uniformities in the light source in the scanner. Usually, the client won't have to correct this, TLA will automatically correct for fixed-pattern anytime a scan is done for a different configuration (film format and resolution). But this should be performed if scanning with the same configuration for two hours or more. There must be NO film in the scanner.

In this version of TLA, corrections for fixed pattern and gain & offset are done automatically every time a scan is requested with a different configuration (resolution and film format) than the last scan. In addition, automatic corrections will be performed after a set time interval, as determined by the *iDarkPointCorrectIntervalMinutes* parameter (set with PutScannerInfo000). If this time period expires, then the fixed pattern and gain & offset corrections will automatically take place when the next scan is requested. This timer is reset whenever a scan with a new configuration is requested, so it will be used only when scanning at the same configuration for long periods.

It is recommended this method be called once upon startup to prepare the scanner. Call it such that the *iCalibrateControl* parameter includes the options CALIBRATE_ExerciseSteppers and CALIBRATE_LampWarmUp. Call it with any resolution and film type, and right after the call to InitializeScanner.

This is a long scan operation. This method cannot be called if another long scan operation is in progress, an error will occur. Once started, the client will get a progress message through the callback interface when complete. However, the client can stop the corrections prematurely by calling ScanCancel.

Callbacks

Operation	Status	Comment
WTO_CorrectionsProgress	WTP_Initialize	Always, right after WT startup
WTO_LampWarmupProgress	WTP_ProgressStart	If requested and wait required
WTO_LampWarmupProgress	WTP_ProgressComplete	If requested and wait required
WTO_ExerciseSteppersProgress	WTP_ProgressStart	If client requested
WTO_ExerciseSteppersProgress	WTP_ProgressComplete	If client requested
WTO_CorrectionsProgress	WTP_ProgressStart	Always
WTO_CorrectionsProgress	WTP_ProgressComplete	If successful
WTO_CorrectionsError	WTP_ProgressComplete	If error
WTO_LampWarmupError	WTP_ProgressComplete	If error with lamp
WTO_ExerciseSteppersError	WTP_ProgressComplete	If error with steppers

See Also

InitializeScanner ScanCancel Awake PutScannerInfo000

3.4.8 ForceDiagnostics

The IScanPictures::ForceDiagnostics method is NOT YET IMPLEMENTED. It will force the scanner to run all its diagnostic programs and report any errors or warnings.

```
HRESULT ForceDiagnostics();
```

Parameters

Return Value

E_NOTIMPL.

Additional Error Codes

Remarks

Example

See Also

3.4.9 GetFilmGuidePosition

The IScanPictures::GetFilmGuidePosition method retrieves the current film guide position.

```
HRESULT GetFilmGuidePosition(int *piFilmFormat);
```

Parameters

piFilmFormat
: [out] The film format (35mm or 24mm) for which the film guide is positioned. See the FILM_FORMAT_000 enumeration for possible values.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists

Remarks

Use PutFilmGuidePosition to change the current position. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

PutFilmGuidePosition

3.4.10 GetScannerInfo000

The IScanPictures::GetScannerInfo000 method retrieves information about the scanner. Mainly operating parameters are retrieved. Use PutScannerInfo000 to change values.

```
HRESULT GetScannerInfo000(int *piScannerType,
                          BSTR *pbstrRomVersion,
                          BSTR *pbstrScannerModel,
                          int *piScannerSerialNumber,
                          int *piScannerVersionHw,
                          BSTR *pbstrTLAVersion,
                          BSTR *pbstrSaveToSharedMemoryEventName,
                          BSTR *pbstrSaveToSharedMemoryMapName,
                          int *piDarkPointCorrectIntervalMinutes,
                          int *piColorPortraitMode,
                          int *pi_uiScanPacketReadyTimeOut,
                          int *pi_uiNoFilmTimeOut,
                          int *piLampSaverSec);
```

Parameters

piScannerType

[out] The type of scanner. See the SCANNER_TYPE_000 enumeration for a list of possible values.

pbstrRomVersion

[out] The versions of wares loaded into the scanner's ROM. This will be in the form "USBmajor.USBminor,CCDhw,CCDsw,LampHw,LampSw,DXhw,DXsw,MotorHw, MotorSw,APSHw,APSSw". The values for "Hw" and "Sw" will be decimal.

pbstrScannerModel

[out] The model of the scanner. This will be "F235" or "F235C".

piScannerSerialNumber

[out] The serial number of the scanner.

piScannerVersionHw

[out] The version of the scanner hardware. See the SCANNER_VERSION_HW_000 enumeration for possible values.

pbstrTLAVersion

[out] The version of the TLA COM server software.

pbstrSaveToSharedMemoryEventName

[out] This parameter is not yet implemented.

pbstrSaveToSharedMemoryMapName

[out] This parameter is not yet implemented.

piDarkPointCorrectIntervalMinutes

[out] The number of minutes between the last correction and the next automatic correction. There is no limit to the range. If this time interval expires, fixed pattern and gain & offset corrections will be done automatically the next time a scan is requested.

piColorPortraitMode

[out] This parameter is not yet implemented.

pi_uiScanPacketReadyTimeOut

[out] The number of milliseconds the COM Server will wait for a packet of scan data. The possible range is 0 to INFINITE.

pi_uiNoFilmTimeOut

[out] The number of seconds the scanner will wait for film once a scan has been started. The possible range is 10 to 300.

piLampSaverSec

[out] This parameter is not yet implemented.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists

Remarks

Some of these parameters can be changed using PutScannerInfo000. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

PutScannerInfo000 InitializeScanner

3.4.11 GetScannerInfo001

The IScanPictures::GetScannerInfo001 method retrieves information about the scanner. Mainly memory parameters are retrieved. Use PutScannerInfo001 to change these parameters.

```
HRESULT GetScannerInfo001(int *pi_uiRingTailDriverBytes,
                          int *pi_uiDriverTriggerBytes,
                          int *pi_uiRingTailProcessedBytes,
                          int *pi_uiProcessedTriggerBytes,
                          int *pi_uiMaxMemoryUsed,
                          int *pi_uiMinMemoryUsed,
                          int *piMaxFilmLength_mm);
```

Parameters

pi_uiRingTailDriverBytes
[out] Undocumented parameter.

pi_uiDriverTriggerBytes
[out] Undocumented parameter.

pi_uiRingTailProcessedBytes
[out] Undocumented parameter.

pi_uiProcessedTriggerBytes
[out] Undocumented parameter.

pi_uiMaxMemoryUsed
[out] The maximum number of bytes of RAM that TLA will use (during a scan).

pi_uiMinMemoryUsed
[out] The minimum number of bytes of RAM that TLA will use (when not scanning).

piMaxFilmLength_mm
[out] The maximum length of a film strip that the scanner will scan before stopping.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists

Remarks

All of these parameters can be changed using PutScannerInfo001. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

PutScannerInfo001

3.4.12 GetScannerInfo002

The IScanPictures::GetScannerInfo002 method retrieves information about the scanner. Mainly scanning information is retrieved. This method is NOT YET IMPLEMENTED.

```
HRESULT GetScannerInfo002(int *piRollIndex,
                          int *piStripIndex,
                          int *piNumberOfPictures,
                          int *piFromFileFrameNumber);
```

Parameters

piRollIndex
[out]

piStripIndex
[out]

piNumberOfPictures
[out]

piFromFileFrameNumber
[out]

Return Value

E_NOTIMPL

Additional Error Codes

Remarks

None of these parameters can be changed. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

3.4.13 GetScannerInfoPreFrame

The IScanPictures::GetScannerInfoPreFrame method retrieves information about the scanner. Both pre-scan framing and cropping parameters are retrieved for a given resolution and film format.

```
HRESULT GetScannerInfoPreFrame(int iResolution,
                               int iFilmFormat,
                               int *piHeightLR,
                               int *piHeightHR_mm,
                               int *piHeightHR,
                               int *piWidthHR,
                               int *piWidthUnitHR,
                               long *plCropHRLeft,
                               long *plCropHRTop,
                               long *plCropHRRight,
                               long *plCropHRBottom);
```

Parameters

iResolution

[in] The scan resolution for which information is requested. Valid values can be found in the RESOLUTION_000 enumeration.

iFilmFormat

[in] The format of film for which information is requested. Valid values can be found in the FILM_FORMAT_000 enumeration.

piHeightLR

[out] The height of the low-resolution buffer frame in pixels. This will depend on the resolution and film format. See the FRAME_SIZES_000 enumeration for possible values.

piHeightHR_mm

[out] The height of the portion of film that will get scanned in 1000's of mm. This will be 23700 for 35mm film and 16200 for 24mm film.

piHeightHR

[out] The height of the high-resolution buffer frame in pixels. This will depend on the resolution and film format. See the FRAME_SIZES_000 enumeration for possible values.

piWidthHR

[out] The width of the high-resolution buffer frame in pixels. This will depend on the resolution and film format. See the FRAME_SIZES_000 enumeration for possible values.

piWidthUnitHR

[out] The distance between the start of one high-resolution buffer frame and the start of the next in pixels.

plCropHRLeft

[out] The distance between the left edge of the high-resolution buffer frame and the left edge of the cropping rectangle in pixels.

plCropHRTop

[out] The distance between the top edge of the high-resolution buffer frame and the top edge of the cropping rectangle in pixels.

plCropHRRight

[out] The distance between the left edge of the high-resolution buffer frame and the right edge of the cropping rectangle in pixels.

plCropHRBottom

[out] The distance between the top edge of the high-resolution buffer frame and the bottom edge of the cropping rectangle in pixels.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter

Remarks

This method retrieves default framing and cropping information that the scanner will be using. To get or set parameters the client wants the scanner to use (for panoramic or half-frame pictures, etc.), use the GetScannerInfoPreFrameUser and PutScannerInfoPreFrameUser methods.

This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

GetScannerInfoPreFrameUser PutScannerInfoPreFrameUser

3.4.14 GetScannerInfoPreFrameUser

The IScanPictures::GetScannerInfoPreFrameUser method retrieves information about the scanner. All user-settable framing and cropping parameters are retrieved for a given resolution and film format.

```
HRESULT GetScannerInfoPreFrameUser(int iResolution,
                                   int iFilmFormat,
                                   int *piWidthHR,
                                   int *piWidthUnitHR,
                                   long *plCropHRLeft,
                                   long *plCropHRTop,
                                   long *plCropHRRight,
                                   long *plCropHRBottom);
```

Parameters

iResolution

[in] The scan resolution for which information is requested. Valid values can be found in the RESOLUTION_000 enumeration.

iFilmFormat

[in] The format of film for which information is requested. Valid values can be found in the FILM_FORMAT_000 enumeration.

piWidthHR

[out] The width of the user-defined high-resolution buffer frame in pixels. This will depend on the resolution and film format. See the FRAME_SIZES_000 enumeration for possible values.

piWidthUnitHR

[out] The distance between the start of one user-defined high-resolution buffer frame and the start of the next in pixels.

plCropHRLeft

[out] The distance between the left edge of the user-defined high-resolution buffer frame and the left edge of the user-defined cropping rectangle in pixels.

plCropHRTop

[out] The distance between the top edge of the user-defined high-resolution buffer frame and the top edge of the user-defined cropping rectangle in pixels.

plCropHRRight

[out] The distance between the left edge of the user-defined high-resolution buffer frame and the right edge of the user-defined cropping rectangle in pixels.

plCropHRBottom

[out] The distance between the top edge of the user-defined high-resolution buffer frame and the bottom edge of the user-defined cropping rectangle in pixels.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter

Remarks

This method retrieves user-defined framing and cropping information that the scanner will be using. Use the PutScannerInfoPreFrameUser method to set these values. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

PutScannerInfoPreFrameUser GetScannerInfoPreFrame

3.4.15 ImportFromFile

The IScanPictures::ImportFromFile method imports picture(s) from file(s) and adds them to the scan group as a separate roll.

```
HRESULT ImportFromFile(BSTR bstrImportFileNames
                       BOOL bDeleteImportOnRelease);
```

Parameters

bstrImportFileNames

[in] A list of the files to open, as a string. Multiple files are specified by separating the filenames with "\n". Each filename should include the full path.

bDeleteImportOnRelease

[in] If TRUE, then the import file(s) will be deleted when the corresponding pictures are released from the save group (by calling ReleaseSaveGroup) or deleted from the scan group (by calling DeleteRollInScanGroup).

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_ScannerNotInitialized EC_TimeOut
EC_UnableToCreateWorkerThread EC_WorkerThreadStartTimeout
EC_FileNameListEmpty EC_TooManyRolls EC_FileNotFound
EC_NoPicturesOrStrips EC_ImportedFileColor EC_MemoryNew
ERROR_CODES_020 EC_PFS_WritePastEOF EC_WIN_VirtualAlloc
EC_WIN_ResetEvent EC_WIN_SetEvent EC_WIN_WaitForSingleObject
EC_WIN_SetFilePointerEx

Remarks

This is a long scan operation. This method cannot be called if another long scan operation is in progress, an error will occur. Once started, the client will get progress messages through the callback interface until complete. However, the client can stop the file import prematurely by calling ScanCancel.

Callbacks

Operation	Status	Comment
WTO_ImportFromFileProgress	WTP_Initialize	Always, right after WT startup
WTO_ImportFromFileProgress	WTP_ProgressStart	Always
WTO_ImportFromFileProgress	WTP_ProgressComplete	If successful
WTO_ImportFromFileError	WTP_ProgressComplete	If error

See Also

ScanCancel Awake ReleaseSaveGroup DeleteRollInScanGroup

3.4.16 LampManualControl

The IScanPictures::LampManualControl method changes the lamp intensity level.

```
HRESULT LampManualControl(int iFilmColor);
```

Parameters

iFilmColor

[in] An integer representing the desired intensity level. Valid values are FILM_COLOR_LAMP_OFF, FILM_COLOR_LAMP_STANDBY, and FILM_COLOR_NEGATIVE from the FILM_COLOR_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter EC_WIN_DeviceIoControl
EC_WIN_ResetEvent EC_TimeOut EC_WIN_WaitForSingleObject
EC_WIN_GetOverlappedResult EC_WIN_FileOpen EC_WIN_FileClose
EC_DRV_PacketBusy EC_DRV_PacketHostError* EC_DRV_InvalidPacketType
EC_DRV_PacketCmdErr EC_DRV_PacketCommErr

Remarks

This method cannot be called if a long scan operation is in progress, an error will occur.

FILM_COLOR_LAMP_OFF will turn the lamp off, FILM_COLOR_NEGATIVE will turn the lamp on (appropriate for any type of scan), and FILM_COLOR_LAMP_STANDBY will dim the lamp. Use FILM_COLOR_LAMP_STANDBY between scans to save the life of the lamp. Because the lamp could be in standby, it is highly recommended to always wait for lamp warmup when calling ForceCorrections or ScanPictures.

See Also

ForceCorrections ScanPictures

3.4.17 PutFilmGuidePosition

The IScanPictures::PutFilmGuidePosition method moves the film guide to an appropriate position for a certain film format.

```
HRESULT PutFilmGuidePosition(int iFileFormat);
```

Parameters

iFileFormat
[in] The film format for which to set the position. See the FILM_FORMAT_000 enumeration for valid values.

Return Value

If successful, S_OK is returned, otherwise an error code will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter EC_ScannerNotInitialized
EC_UnableToCreateWorkerThread EC_WorkerThreadClientSignal
EC_WorkerThreadStartTimeout EC_PreviousError EC_StepperAlreadyMoving
EC_TimeOut EC_WIN_GetOverlappedResult EC_WIN_DeviceIoControl
EC_WIN_ResetEvent EC_WIN_WaitForSingleObject EC_WIN_FileOpen
EC_DRV_PacketBusy EC_DRV_InvalidPacketType EC_DRV_PacketCmdErr
EC_DRV_PacketCommErr EC_DRV_PacketHostError*

Remarks

Make sure there is no film in the scanner prior to calling this function. The current film guide position can be retrieved using GetFilmGuidePosition.

This is a long scan operation. This method cannot be called if another long scan operation is in progress, an error will occur. Once started, the client will get progress messages through the callback interface until complete. However, the client can stop the film guide positioning prematurely by calling ScanCancel.

Callbacks

Operation	Status	Comment
WTO_PutFilmGuidePositionProgress	WTP_Initialize	Always, right after WT startup
WTO_PutFilmGuidePositionProgress	WTP_ProgressStart	Always
WTO_PutFilmGuidePositionProgress	WTP_ProgressComplete	If successful
WTO_PutFilmGuidePositionError	WTP_ProgressComplete	If error

See Also

GetFilmGuidePosition Awake ScanCancel

3.4.18 PutFilmPressureRollerPosition

The IScanPictures::PutFilmPressureRollerPosition method engages or disengages the film pressure rollers.

```
HRESULT PutFilmPressureRollerPosition(BOOL bEngage);
```

Parameters

bEngage
[in] False to disengage the rollers, True to engage them.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter EC_ScannerNotInitialized
EC_UnableToCreateWorkerThread EC_WorkerThreadStartTimeout EC_TimeOut
EC_MemoryNew EC_WIN_FileOpen EC_WIN_WaitForSingleObject
EC_WIN_ResetEvent EC_WIN_DeviceIoControl EC_WIN_GetOverlappedResult
EC_WIN_SetEvent EC_DRV_PacketHostError* EC_DRV_InvalidPacketType
EC_DRV_PacketCmdErr EC_DRV_PacketCommErr EC_DRV_PacketBusy

Remarks

Use this function to disengage the film pressure rollers in case the film is jammed and the operator cannot remove it using AdvanceFilm. It is not necessary to reengage the pressure rollers, they will reengage automatically when another scan is started.

This is a long scan operation. This method cannot be called if another long scan operation is in progress, an error will occur. Once started, the client will get progress messages through the callback interface when complete. However, the client can stop the pressure roller positioning prematurely by calling ScanCancel.

Callbacks

Operation	Status	Comment
WTO_PutFilmPressureRollersPositionProgress	WTP_Initialize	Always, after WT startup
WTO_PutFilmPressureRollersPositionProgress	WTP_ProgressStart	Always
WTO_PutFilmPressureRollersPositionProgress	WTP_ProgressComplete	If successful
WTO_PutFilmPressureRollersPositionError	WTP_ProgressComplete	If error

See Also

Awake ScanCancel

3.4.19 PutScannerInfo000

The IScanPictures::PutScannerInfo000 method changes information about the scanner. Mainly operating parameters are changed. Use GetScannerInfo000 to get the current values.

```
HRESULT PutScannerInfo000(int iDarkPointCorrectIntervalMinutes,
                          int iColorPortraitMode,
                          int i_uiScanPacketReadyTimeOut,
                          int i_uiNoFilmTimeOut,
                          int iLampSaverSec);
```

Parameters

iDarkPointCorrectIntervalMinutes

[in] The number of minutes between the last correction and the next automatic correction (see GetScannerInfo000). The valid range is 1 to 525600. The default value is 120.

iColorPortraitMode

[in] Indicates whether color portrait mode is used or not. For valid values, see the COLOR_PORTRAIT_MODE_000 enumeration. COLOR_PORTRAIT_MODE_NOT is the default value. This feature is NOT YET IMPLEMENTED.

i_uiScanPacketReadyTimeOut

[in] The number of milliseconds the COM Server will wait for a packet of scan data. If this time expires, TLA will send an EC_TimeOut error. The valid range is 0 to INFINITE. The default value is 1000.

i_uiNoFilmTimeOut

[in] The number of seconds the scanner will wait for film once a scan has been started. If this time expires, TLA will send an EC_NoFilmTimeOut error. The valid range is 10 to 5 X 60. The default value is 60.

iLampSaverSec

[in] This parameter is not yet implemented. The default value is 60.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter

Remarks

The current value of these parameters can be retrieved using GetScannerInfo000. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

GetScannerInfo000 InitializeScanner

3.4.20 PutScannerInfo001

The IScanPictures::PutScannerInfo001 method changes information about the scanner. Mainly memory parameters are changed. Use GetScannerInfo001 to get current values.

```
HRESULT PutScannerInfo001(int i_uiRingTailDriverBytes,
                          int i_uiDriverTriggerBytes,
                          int i_uiRingTailProcessedBytes,
                          int i_uiProcessedTriggerBytes,
                          int i_uiMaxMemoryUsed,
                          int i_uiMinMemoryUsed,
                          int iMaxFilmLength_mm);
```

Parameters

i_uiRingTailDriverBytes
[in] Undocumented parameter. The default value is 0x800000.

i_uiDriverTriggerBytes
[in] Undocumented parameter. The default value is 0x100000.

i_uiRingTailProcessedBytes
[in] Undocumented parameter. The default value is 0x1600000.

i_uiProcessedTriggerBytes
[in] Undocumented parameter. The default value is 0xB00000.

i_uiMaxMemoryUsed
[in] The maximum number of bytes of RAM that TLA will use (during a scan). The valid range is 8 Mb to 256 Mb. The default value is 0x2600000.

i_uiMinMemoryUsed
[in] The minimum number of bytes of RAM that TLA will use (when not scanning). The valid range is 2 Mb to *i_uiMaxMemoryUsed*. The default value is 0x300000.

iMaxFilmLength_mm
[in] The maximum length of a film strip that the scanner will scan before stopping. The valid range is 24 to 6400. The default value is 1600 mm (~63 in).

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_WIN_SetProcessWorkingSetSize

Remarks

The current value of these parameters can be retrieved using GetScannerInfo001. This method cannot be called if a long scan operation is in progress, an error will occur. If the maximum film length is increased, then the registry setting "HKEY_LOCAL_MACHINE\SOFTWARE\Pakon\TLA\Scan\HiResMegabytesRoll" must also be increased. The formula is HiResMegabytesRoll = MaxFilmLengthmm * 1350211 + extra, where extra is 100 megabytes or so (this is because the disk space used will be rounded up to an even number of sectors).

See Also

GetScannerInfo001

3.4.21 PutScannerInfoPreFrameUser

The IScanPictures::PutScannerInfoPreFrameUser method changes information about the scanner. All user-settable framing and cropping parameters can be changed for a given resolution and film format. Use this method for custom framing and cropping, for example, if the film is panoramic, half-frame, wide-lux or non-perforated.

```
HRESULT PutScannerInfoPreFrameUser(int iResolution,
                                   int iFilmFormat,
                                   int iWidthHR,
                                   int iWidthUnitHR,
                                   long lCropHRLeft,
                                   long lCropHRTop,
                                   long lCropHRRight,
                                   long lCropHRBottom);
```

Parameters

iResolution

[in] The scan resolution for which information is to be changed. Valid values can be found in the RESOLUTION_000 enumeration.

iFilmFormat

[in] The format of film for which information is to be changed. Valid values can be found in the FILM_FORMAT_000 enumeration.

iWidthHR

[in] The width of the user-defined high-resolution buffer frame in pixels. This will depend on the resolution and film format. See the FRAME_SIZES_000 enumeration for possible values.

iWidthUnitHR

[in] The distance between the start of one user-defined high-resolution buffer frame and the start of the next in pixels. See Remarks.

lCropHRLeft

[in] The distance between the left edge of the user-defined high-resolution buffer frame and the left edge of the user-defined cropping rectangle in pixels. See Remarks.

lCropHRTop

[in] The distance between the top edge of the user-defined high-resolution buffer frame and the top edge of the user-defined cropping rectangle in pixels. See Remarks.

lCropHRRight

[in] The distance between the left edge of the user-defined high-resolution buffer frame and the right edge of the user-defined cropping rectangle in pixels. See Remarks.

lCropHRBottom

[in] The distance between the top edge of the user-defined high-resolution buffer frame and the bottom edge of the user-defined cropping rectangle in pixels. See Remarks.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter

Remarks

This method sets user-defined framing and cropping information that the scanner should use. Once called, the scanner will use these values. To convert back to the default values, call GetScannerInfoPreFrame to get the default values, then call PutScannerInfoPreFrameUser with these values. Use the GetScannerInfoPreFrameUser method to get the current user-defined values. The low-resolution parameters will be automatically set according to the LR/HR ratio.

The cropping rectangle defined by the above parameters must fit inside the high-resolution buffer frame and be no smaller than 1/20 of the width and 1/20 of the height. *iWidthUnitHR* must be at least ¼ of the default unit width (*piWidthUnitHR* from GetScannerInfoPreFrame) and at least *iWidthHR* + 25 and less than the maximum film length (converted to pixels).

This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

GetScannerInfoPreFrameUser GetScannerInfoPreFrame

3.4.22 ResetFactoryDefaults

The IScanPictures::ResetFactoryDefaults method will reset the focus and/or motor speed adjustments to the factory default.

```
HRESULT ResetFactoryDefaults(int iFactoryResetControl,
                             int iResolution,
                             int iFilmFormat);
```

Parameters

iFactoryResetControl

[in] The operation for which parameters are to be reset. Valid values can be found in the FACTORY_RESET_CONTROL_000 enumeration. Motor speed and focus can be reset at the same time by summing values.

iResolution

[in] The scan resolution for which parameters are to be reset. Valid values can be found in the RESOLUTION_000 enumeration.

iFilmFormat

[in] The film format for which parameters are to be reset. Valid values can be found in the FILM_FORMAT_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_ScannerNotInitialized EC_InvalidParameter

Remarks

Calling this function will undo the effects of any calls to the AdjustMotorSpeed function or the ForceCorrections function (for focus) or both. It will return these settings to the values established at the time the scanner was manufactured. This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

AdjustMotorSpeed ForceCorrections

3.4.23 ResetStatusLeds

The IScanPictures::ResetStatusLeds method will reset the scanner's LEDs to their initial state.

```
HRESULT ResetStatusLeds();
```

Parameters

None.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_ScannerNotInitialized EC_WIN_SetEvent

Remarks

When errors occur, the LEDs may be turned yellow or red. Calling this function will reset these LEDs to their initial state (off or green). This method cannot be called if a long scan operation is in progress, an error will occur.

See Also

3.4.24 ScanCancel

The IScanPictures::ScanCancel method will stop any long scan operation currently running.

```
HRESULT ScanCancel();
```

Parameters

None.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WIN_SetEvent

Remarks

Calling this function will stop any long scan operation currently running, but it won't reverse the progress of the operation. For example, if scanning, this function will stop the scan, but any images already in the buffers will remain. If advancing film, this function will stop the advance, but won't reverse the film to its original location. If forcing corrections, this function will stop any further corrections, but won't undo the corrections that have already completed.

See Also

AdvanceFilm FilmTrackTest ForceCorrections
PutFilmGuidePosition PutFilmPressureRollerPosition ScanPictures

3.4.25 ScanPictures

The IScanPictures::ScanPictures method scans pictures. It tells the scanner about the characteristics of the film about to be scanned, then starts the scan process.

```
HRESULT ScanPictures(int iResolution,
                     int iFilmColor,
                     int iFilmFormat,
                     int iStripMode,
                     int iScanControl);
```

Parameters

iResolution

[in] The resolution at which the client wants to scan (base4, base8, etc.). Valid values can be found in the RESOLUTION_000 enumeration.

iFilmColor

[in] The type of film to be scanned (color, black&white, etc.). Valid values can be found in the FILM_COLOR_000 enumeration.

iFilmFormat

[in] The format of the film to be scanned (24mm, 35mm, etc.). Valid values can be found in the FILM_FORMAT_000 enumeration.

iStripMode

[in] The strip mode to use for this scan (full roll | single strip | multiple strips). Valid values can be found in the STRIP_MODE_000 enumeration.

iScanControl

[in] The bitwise sum of controls the client wants to use for this scan. Valid values can be found in the SCAN_CONTROL_000 enumeration. It is recommended to always include the lamp warmup option.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_BufferDriveMegabytesRollTooSmall EC_EEPromAddress EC_EEPromLength
EC_EEPromMemoryAddress EC_FileNotFound EC_FocusCurvatureThreshold
EC_FocusPredictorThreshold EC_FocusOutsideRegionOfInterest EC_LampError
EC_FocusQuadRegress EC_HardwareFault EC_InvalidMemberVariable
EC_InvalidParameter EC_MemoryNew EC_NoFixedPatternCorrection
EC_NoHighResolutionBuffer EC_NoStripsScanned EC_BadFileData
EC_WorkerThreadClientSignal EC_WorkerThreadExists
EC_WorkerThreadStartTimeout

EC_PFS_FilePointerDeleted EC_PFS_InvalidPointer EC_PFS_NullFilePointer
EC_PFS_PartitionSelected EC_PFS_WritePastEOF EC_PFS_WriteSizeInvalid
EC_PFS_WritingToCompletedStrip ERROR_CODES_010 ERROR_CODES_020

EC_ApsFilmJamExtract EC_ApsFilmJamScan EC_ApsFilmJamRetract
EC_ApsPark EC_ApsParkInit EC_ProcessedRingTailOverflow
EC_QueryInterface EC_ScanLineAcquisition EC_ScannerNotInitialized
EC_TimeOut EC_StepperAlreadyMoving EC_StepperDidNotStop
EC_TooManyRolls EC_UnableToCreateWorkerThread EC_NoFilmTimeOut
EC_BadSimulatorFile EC_PreviousHardwareFaultAps

EC_WIN_CancelIo EC_WIN_CreateEvent EC_WIN_DeviceIoControl
EC_WIN_FileOpen EC_WIN_FileRead EC_WIN_FileWrite
EC_WIN_GetOverlappedResult EC_WIN_ResetEvent EC_WIN_SetEvent
EC_WIN_SetFilePointerEx EC_WIN_SetProcessWorkingSetSize
EC_WIN_VirtualAlloc EC_WIN_VirtualFree EC_WIN_VirtualLock
EC_WIN_VirtualUnlock EC_WIN_WaitForSingleObject

Remarks

This is a long scan operation. This method cannot be called if another long scan operation is in progress, an error will occur. If scanning in strip mode, the client will get (absolute) progress messages through the callback interface until complete (one message for each strip). The client can stop the scan prematurely by calling ScanCancel.

Once scanned, the pictures are retained in the scan group as a roll. At most 24 rolls can be retained in the scan group at any given time (this may be further limited by the registry settings of HiResMegabytesRoll and HiResMegabytesTotal). Rolls can be moved from the scan group to the save group using MoveOldestRollToSaveGroup.

For 24mm film, iFilmColor cannot be set to FILM_COLOR_POSITIVE, strip mode cannot be used, and if SCAN_Use24mmAutoLoaderMOF is specified, then SCAN_Use24mmAutoLoader must be specified. For 35mm film, neither SCAN_Use24mmAutoLoaderMOF nor SCAN_Use24mmAutoLoader can be specified. For 24mm or 35mm film, if iFilmColor is set to FILM_COLOR_BnW_NORMAL, then scratch removal cannot be used.

Callbacks

Operation	Status	Comment
WTO_ScanProgress	WTP_Initialize	Always, right after WT startup
WTO_LampWarmupProgress	WTP_ProgressStart	If requested and wait required
WTO_LampWarmupProgress	WTP_ProgressStart + x	< WTP_ProgressEnd
WTO_LampWarmupProgress	WTP_ProgressComplete	If requested and wait required
WTO_CorrectionsProgress	WTP_ProgressStart	If fixed pattern update required
WTO_CorrectionsProgress	WTP_ProgressComplete	If fixed pattern update required
WTO_ScanProgress	WTP_ProgressStart	Always
WTO_ScanProgress	WTP_ProgressStart + x	< WTP_ProgressEnd, strip mode
WTO_ScanProgress	WTP_ProgressComplete	If successful
WTO_ScanError	WTP_ProgressComplete	If error
WTO_LampWarmupError	WTP_ProgressComplete	If error
WTO_CorrectionsError	WTP_ProgressComplete	If error

See Also

ScanCancel MoveOldestRollToSaveGroup Awake

3.4.26 StopFilmDrive

The IScanPictures::StopFilmDrive method is not yet fully implemented. Calling this function is the same as calling ScanCancel.

```
HRESULT StopFilmDrive();
```

Parameters

None.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WIN_SetEvent

Remarks

See ScanCancel.

See Also

ScanCancel

3.5 Save Pictures

The functions in this section all concern processing and saving pictures. Some of the operations take a while to complete and so these operations get their own processing thread. These are called “long save operations” and are marked as such in the function description. When these functions are called, they start their own processing thread, then return immediately.

The long save operations include:

SaveToClientMemory SaveToDisk

When a long save operation is running, the client cannot call any other function from the Save Pictures interface with the exception of the ClientMemoryBufferAdd function and the SaveCancel function. However, the client can call any function from the Scan Pictures interface.

The following is a list of all the Save Pictures functions in alphabetical order.

3.5.1 ClientMemoryBufferAdd

The ISavePictures::ClientMemoryBufferAdd method informs TLA about a client memory area where pictures will be saved using SaveToClientMemory. TLA adds this buffer to its list of client buffers. It is the responsibility of the client to allocate and deallocate each buffer.

```
HRESULT ClientMemoryBufferAdd(int i_pByteStartPointer,
                              int iByteCount);
```

Parameters

i_pByteStartPointer
[in] A pointer to the start of the client memory area.

iByteCount
[in] The number of bytes set aside for this buffer.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_MemoryNew EC_WIN_SetEvent

Remarks

This method can be called anytime. The `iRequiredBufferSize` function from the TLAClientDemo illustrates how to calculate the size of buffer needed for each picture. More than one buffer can be added to TLA, but the client is responsible for allocating and deallocating each buffer. Each buffer must be allocated prior to calling this function, and each buffer must be removed from TLA's list prior to deallocating. As buffers are filled by TLA, they are removed automatically from TLA's list. To remove all buffers, call ClientMemoryBufferDismissAll.

Example

```
m_iBufferSize = iRequiredBufferSize(csSaveSize,
                                    iSaveToMemoryFormat);
if(m_pClientBuffer1 = new UCHAR[m_iBufferSize])
{
   hr = m_pISavePictures->ClientMemoryBufferAdd(
                                          (int)m_pClientBuffer1,
                                          m_iBufferSize);
}
```

See Also

SaveToClientMemory ClientMemoryBufferDismissAll

3.5.2 ClientMemoryBufferDismissAll

The ISavePictures::ClientMemoryBufferDismissAll method removes all the (unused) client buffers from TLA's list of buffers. These buffers were added using ClientMemoryBufferAdd.

```
HRESULT ClientMemoryBufferDismissAll();
```

Parameters

None

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_ClientMemoryBufferInUse EC_WIN_ResetEvent

Remarks

This method can be called anytime.

See Also

ClientMemoryBufferAdd

3.5.3 DeletePicture

The ISavePictures::DeletePicture method deletes a picture from the list of pictures in the save group.

```
HRESULT DeletePicture(int iIndex);
```

Parameters

iIndex

[in] The index of the picture the client wants to delete. Valid values are 0 to the number of pictures in the save group – 1, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidParameter EC_NoPicturesOrStrips
EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur. When a picture is deleted, the other pictures with higher indexes are shifted down one. That way, the picture indexes always range from 0 to the number of pictures in the save group – 1.

See Also

InsertPicture

3.5.4 DeleteRollInScanGroup

The ISavePictures::DeleteRollInScanGroup method deletes a specified roll from the group of scanned rolls.

```
HRESULT DeleteRollInScanGroup(int iRollIndex);
```

Parameters

iRollIndex

[in] The index of the roll to be deleted. Valid values are 0 to the number of rolls in the scan group - 1. (0 is the oldest roll, and #rolls - 1 is the newest roll.)

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidIndex

Remarks

Calling this method will permanently delete the scanned images, they cannot be retrieved. An error will occur if there are no rolls in the scan group. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

ScanPictures

3.5.5 GetPictureColorSettings

The ISavePictures::GetPictureColorSettings method retrieves color information about a certain picture in the save group.

```
HRESULT GetPictureColorSettings(int iIndex,
                                int *piRed,
                                int *piGreen,
                                int *piBlue,
                                int *piBrightness,
                                int *piContrast,
                                int *piSharpness);
```

Parameters

iIndex
[in] The index of the picture for which settings are desired. Valid values are zero to one less than the number of pictures, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piRed
[out] The amount of red saturation in this picture. The possible range is –1000 to 1000.

piGreen
[out] The amount of green saturation in this picture. The possible range is –1000 to 1000.

piBlue
[out] The amount of blue saturation in this picture. The possible range is –1000 to 1000.

piBrightness
[out] The brightness setting of this picture. The possible range is –1000 to 1000.

piContrast
[out] The contrast setting of this picture. The possible range is –1000 to 1000.

piSharpness
[out] The sharpness of the picture. The possible range is –1000 to 1000.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur.

See Also

PutPictureColorSettings

3.5.6 GetPictureCountSaveGroup

The ISavePictures::GetPictureCountSaveGroup method retrieves information about the rolls of film in the save group.

```
HRESULT GetPictureCountSaveGroup(int *piRollCount,
                                 int *piStripCount,
                                 int *piPictureCount,
                                 int *piPictureSelectedCount,
                                 int *piPictureHiddenCount);
```

Parameters

piRollCount
[out] The number of rolls in the save group.

piStripCount
[out] The total number of film strips in the save group.

piPictureCount
[out] The total number of pictures in the save group.

piPictureSelectedCount
[out] The total number of pictures that are marked as “selected”.

piPictureHiddenCount
[out] The total number of pictures that are marked as “hidden”.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur.

See Also

3.5.7 GetPictureCountScanGroup

The ISavePictures::GetPictureCountScanGroup method retrieves information about a roll of film that has been scanned.

```
HRESULT GetPictureCountScanGroup(int iRollIndex,
                                 int *piStripCount,
                                 int *piPictureCount,
                                 int *piScanWarnings);
```

Parameters

iRollIndex
: [in] The index of the roll for which to get information. Valid values are 0 to the number of rolls in the scan group – 1. (0 is the oldest roll, and #rolls-1 is the newest roll.)

piStripCount
: [out] The number of film strips associated with this roll.

piPictureCount
: [out] The number of pictures associated with this roll.

piScanWarnings
: [out] The bitwise sum of all warnings that occurred during the scan. Possible values can be found in the SCAN_WARNINGS_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur.

Example

```
hr = m_pIScanPictures->
         GetPictureCountScanGroup(0, &m_iScanStripCount,
                          &m_iScanPictureCount, &m_iScanWarnings);
if(FAILED(hr))
{
   ::AnalyzeComError(hr, IID_IScanPictures,
                     m_pIScanPictures, NULL);
}
else if (m_iScanWarnings != 0)
{
   CheckScanWarnings(m_iScanWarnings);
}
```

3.5.8 GetPictureFramingInfo

The ISavePictures::GetPictureFramingInfo method retrieves information about the original framing parameters of a certain picture in the save group, as determined by TLA's framing algorithm.

```
HRESULT GetPictureFramingInfo(int iIndex,
                              int *piFramingRisk,
                              long *plLeftHR,
                              long *plTopHR,
                              long *plRightHR,
                              long *plBottomHR);
```

Parameters

iIndex

[in] The index of the picture for which to retrieve framing information. Valid values are zero to one less than the number of pictures in the save group, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

piFramingRisk

[out] The framing risk for this picture. See the FRAMING_RISK_000 enumeration for possible values. Unless there were scan warnings for framing, this will be very low.

plLeftHR

[out] The left edge of this picture's frame in the high-resolution buffer in pixels.

plTopHR

[out] The top edge of this picture's frame in the high-resolution buffer in pixels.

plRightHR

[out] The right edge of this picture's frame in the high-resolution buffer in pixels.

plBottomHR

[out] The bottom edge of this picture's frame in the high-resolution buffer in pixels.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur. Clients can set their own framing parameters with a call to PutPictureFramingUserInfo. This will override the framing parameters established by the framing algorithm. However, a client can always retrieve the original framing parameters using this function.

See Also

GetScannerInfoPreFrame GetPictureFramingUserInfo PutPictureFramingUserInfo

3.5.9 GetPictureFramingUserInfo

The ISavePictures::GetPictureFramingUserInfo method retrieves information about the user's framing parameters of a certain picture in the save group.

```
HRESULT GetPictureFramingUserInfo(int iIndex,
                                  long *plLeftHR,
                                  long *plTopHR,
                                  long *plRightHR,
                                  long *plBottomHR);
```

Parameters

iIndex

[in] The index of the picture for which to retrieve user framing information. Valid values are zero to one less than the number of pictures in the save group, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

plLeftHR

[out] The left edge of this picture's frame in the high-resolution buffer in pixels. The possible range is zero to 16 less than the length of the buffer.

plTopHR

[out] The top edge of this picture's frame in the high-resolution buffer in pixels. The possible range is zero to 16 less than the height of the buffer.

plRightHR

[out] The right edge of this picture's frame in the high-resolution buffer in pixels. The possible range is 16 to the length of the buffer.

plBottomHR

[out] The bottom edge of this picture's frame in the high-resolution buffer in pixels. The possible range is 16 to one less than the height of the buffer.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur. This function will retrieve the framing parameters set by the client. The client sets the parameters with a call to PutPictureFramingUserInfo. The original framing parameters can be retrieved with a call to GetPictureFramingInfo.

See Also

GetPictureFramingInfo PutPictureFramingUserInfo

3.5.10 GetPictureFramingUserInfoLowRes

The ISavePictures::GetPictureFramingUserInfoLowRes method retrieves information about the user's low-resolution framing parameters of a certain picture in the save group.

```
HRESULT GetPictureFramingUserInfoLowRes(int iIndex,
                                        long *plLeftLR,
                                        long *plTopLR,
                                        long *plRightLR,
                                        long *plBottomLR);
```

Parameters

iIndex
[in] The index of the picture for which to retrieve user framing information. Valid values are zero to one less than the number of pictures in the save group, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

plLeftLR
[out] The left edge of this picture's frame in the low-resolution buffer in pixels.

plTopLR
[out] The top edge of this picture's frame in the low-resolution buffer in pixels.

plRightLR
[out] The right edge of this picture's frame in the low-resolution buffer in pixels.

plBottomLR
[out] The bottom edge of this picture's frame in the low-resolution buffer in pixels.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur. The client's framing parameters are set with a call to PutPictureFramingUserInfo. The high-resolution parameters are retrieved with a call to GetPictureFramingUserInfo. This function will retrieve the low-resolution parameters.

See Also

GetPictureFramingUserInfo PutPictureFramingUserInfo

3.5.11 GetPictureInfo

The ISavePictures::GetPictureInfo method retrieves information about a certain picture in the save group. Use PutPictureInfo to change some of these values.

```
HRESULT GetPictureInfo(int iIndex,
                       int *piRollIndexFromStrip,
                       int *piStripIndexFromStrip,
                       int *piFilmProductFromStrip,
                       int *piFilmSpecifierFromStrip,
                       BSTR *pbstrFrameName,
                       int *piFrameNumber,
                       int *piPrintAspectRatio,
                       BSTR *pbstrFileName,
                       BSTR *pbstrDirectory,
                       int *piRotation,
                       int *piSelectedHidden);
```

Parameters

iIndex

[in] The index of the picture for which information will be retrieved. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piRollIndexFromStrip

[out] The index of the roll to which this picture belongs. Possible values are 0 to the number of rolls in the save group – 1.

piStripIndexFromStrip

[out] The index of the strip to which this picture belongs. Possible values are 0 to the number of strips in the save group – 1.

piFilmProductFromStrip

[out] The strip's film product class. Initially, this is determined from the DX codes. If DX is not read, this will be -1.

piFilmSpecifierFromStrip

[out] The strip's film specifier. Initially, this is determined from the DX codes. If DX is not read, this will be -1.

pbstrFrameName

[out] The name of the frame. This is determined from the frame number and the frame type (print aspect ratio). Typical values are "3A" or "6P". If DX is not read, this will be "DX_Error."

piFrameNumber

[out] The number of the frame. Initially, this is determined from the DX codes. For example, if a roll has 24 frames, then the frame numbers will range from 2 to 49. If DX is not read, then this will be INT_MIN.

piPrintAspectRatio

[out] The type of aspect ratio for the frame. See the FRAME_TYPE_000 enumeration for possible values. Applies only to 24mm cartridge scans. If scanning 35mm or scanning 24mm out of the cartridge, this will be PRINT_ASPECT_RATIO_H.

pbstrFileName

[out] The name of the file where this picture will be saved with SaveToDisk. This parameter is used ONLY for the SaveToDisk method.

pbstrDirectory

[out] The location of the file where this picture will be saved with SaveToDisk. This parameter is used ONLY for the SaveToDisk method.

piRotation

[out] The rotation of the picture. Possible values can be found in the ROTATE_000 enumeration. Initially this will be ROTATE_0.

piSelectedHidden

[out] Indicates whether the picture is hidden or selected. Possible values can be found in the S_OR_H_000 enumeration. Initially this is S_OR_H_NONE.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex
EC_InvalidMemberVariable

Remarks

This method cannot be called if a long save operation is in progress, an error will occur.

See Also

PutPictureInfo SaveToDisk

3.5.12 GetPictureInfo1

The ISavePictures::GetPictureInfo1 method retrieves information about a certain picture in the save group. Use PutPictureInfo1 to change most of these values.

```
HRESULT GetPictureInfo1(int iIndex,
                        BSTR *pbstrFrameName,
                        int *piFrameNumber,
                        BSTR *pbstrFileName,
                        BSTR *pbstrDirectory);
```

Parameters

iIndex

[in] The index of the picture for which information will be retrieved. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

pbstrFrameName

[out] The name of the frame. This is determined from the frame number and the frame type. Typical values are "3A" or "6P". If DX is not read, this will be "DX_Error."

piFrameNumber

[out] The number of the frame as determined from the DX codes. For example, if a roll has 24 frames, then the frame numbers will range from 2 to 49. If DX is not read, then this will be INT_MIN.

pbstrFileName

[out] The name of the file where this picture will be saved with SaveToDisk. This parameter is used ONLY for the SaveToDisk method.

pbstrDirectory

[out] The location of the file where this picture will be saved with SaveToDisk. This parameter is used ONLY for the SaveToDisk method.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

Use PutPictureInfo1 to change most of these values. This method cannot be called if a long save operation is in progress, an error will occur. This method is identical to GetPictureInfo with a few less parameters.

See Also

PutPictureInfo1 SaveToDisk GetPictureInfo

3.5.13 GetPictureInfo2

The ISavePictures::GetPictureInfo2 method retrieves additional information about a certain picture in the save group.

```
HRESULT GetPictureInfo2(int iIndex,
                        int *piMagneticDataStatus,
                        int *piPrintAspectRatio,
                        BSTR *pbstrTitleOrImportFileName,
                        int *piYear,
                        int *piMonth,
                        int *piDayOfMonth,
                        int *piSecondsAfterMidnight);
```

Parameters

iIndex

[in] The index of the picture for which information will be retrieved. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piMagneticDataStatus

[out] The status of the magnetic data from MOF. Possible values can be found in the MAGNETIC_DATA_STATUS_000 enumeration.

piPrintAspectRatio

[out] The type of aspect ratio for the frame. See the FRAME_TYPE_000 enumeration for possible values. Applies only to 24mm cartridge scans. If scanning 35mm or scanning 24mm out of the cartridge, this will be PRINT_ASPECT_RATIO_H.

pbstrTitleOrImportFileName

[out] The title of the picture from MOF, or the filename if this picture was imported.

piYear

[out] The year the picture was taken (from MOF) or the imported file was created.

piMonth

[out] The month the picture was taken (from MOF) or the imported file was created.

piDayOfMonth

[out] The day of the month the picture was taken (from MOF) or the imported file was created.

piSecondsAfterMidnight

[out] The time the picture was taken in seconds after midnight (from MOF) or the imported file was created.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

This method cannot be called if a long save operation is in progress, an error will occur. This method is useful only for pictures from APS film with magnetic data or to get file information about a picture that was imported.

See Also

ScanPictures

3.5.14 GetPictureLightingDirection

The ISavePictures::GetPictureLightingDirection method retrieves information about the direction of light of a certain picture in the save group. Lighting direction is NOT YET IMPLEMENTED.

```
HRESULT GetPictureLightingDirection(int iIndex,
                                    int *piPictureLightingDirection);
```

Parameters

iIndex

[in] The index of the picture for which to retrieve lighting information. Valid values are zero to one less than the number of pictures in the save group, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

piPictureLightingDirection

[out] The direction of light for this picture. Possible values can be found in the PICTURE_LIGHTING_DIRECTION_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

Remarks

The lighting direction can be set with PutPictureLightingDirection and retrieved with this function, but it is not yet implemented in the color correction algorithms.

See Also

PutPictureLightingDirection

3.5.15 GetPictureRedEyeSettings

The ISavePictures::GetPictureRedEyeSettings method retrieves information about the red-eye parameters of a certain picture in the save group. Red eye is NOT YET IMPLEMENTED.

```
HRESULT GetPictureRedEyeSettings(int iIndex,
                                 int *piRedEye,
                                 long *plLeftHR,
                                 long *plTopHR,
                                 long *plRightHR,
                                 long *plBottomHR);
```

Parameters

iIndex

[in] The index of the picture for which to retrieve framing information. Valid values are zero to one less than the number of pictures in the save group, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

piRedEye

[out] The red-eye setting for this picture. See the RED_EYE_000 enumeration for possible values.

plLeftHR

[out] The left edge of this picture's red-eye rectangle in the high-res buffer in pixels.

plTopHR

[out] The top edge of this picture's red-eye rectangle in the high-res buffer in pixels.

plRightHR

[out] The right edge of this picture's red-eye rectangle in the high-res buffer in pixels.

plBottomHR

[out] The bottom edge of this picture's red-eye rectangle in the high-res buffer in pixels.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

Remarks

The red eye settings can be set with PutPictureRedEyeSettings and retrieved with this function, but red eye reduction is not yet implemented.

See Also

PutPictureRedEyeSettings

3.5.16 GetPictureRotation

The ISavePictures::GetPictureRotation method retrieves the rotation parameter of a certain picture in the save group. Use PutPictureRotation to change this value.

```
HRESULT GetPictureRotation(int iIndex,
                           int *piRotation);
```

Parameters

iIndex

[in] The index of the picture for which information will be retrieved. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piRotation

[out] The rotation of the picture. Possible values can be found in the ROTATE_000 enumeration. Initially this will be ROTATE_0.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

Use PutPictureRotation to change this value. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

PutPictureRotation

3.5.17 GetPictureSelection

The ISavePictures::GetPictureSelection method retrieves the selection/hidden parameter of a certain picture in the save group. Use PutPictureSelection to change this value.

```
HRESULT GetPictureSelection(int iIndex,
                            int *piSelectedHidden);
```

Parameters

iIndex

[in] The index of the picture for which information will be retrieved. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piSelectedHidden

[out] Indicates whether the picture is hidden or selected. Possible values can be found in the S_OR_H_000 enumeration. Initially this is S_OR_H_NONE.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

Use PutPictureSelection to change this value. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

PutPictureSelection

3.5.18 GetRollCountScanGroup

The ISavePictures::GetRollCountScanGroup method retrieves the number of rolls currently in the scan group.

```
HRESULT GetRollCountScanGroup(int *piRollCount);
```

Parameters

piRollCount
[out] The number of rolls currently in the scan group.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists

Remarks

This method cannot be called if a long save operation is in progress, an error will occur.

See Also

ScanPictures

3.5.19 GetSaveInfo

The ISavePictures::GetSaveInfo method retrieves information about how pictures will be saved with the SaveToDisk method. Use PutSaveInfo to change these values.

```
HRESULT GetSaveInfo(int *piDefaultFilmProduct,
                    int *piDefaultFilmSpecifier,
                    BSTR *pbstrDefaultDirectory);
```

Parameters

piDefaultFilmProduct
[out] This parameter is not used.

piDefaultFilmSpecifier
[out] This parameter is not used.

pbstrDefaultDirectory
[out] The default directory. Initially this is "c:\temp".

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

Use PutSaveInfo to change these values. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

PutSaveInfo SaveToDisk

3.5.20 GetStripInfo

The ISavePictures::GetStripInfo method retrieves information about a certain strip in the save group.

```
HRESULT GetStripInfo(int iStripIndex,
                     int *piRollIndex,
                     BOOL *pbIsStrip,
                     int *piFilmColor,
                     int *piFilmFormat,
                     long *plHeightHR,
                     long *plLengthHR,
                     long *plHeightLR,
                     long *plLengthLR,
                     int *piScanWarnings,
                     int *piFilmProduct,
                     int *piFilmSpecifier,
                     int *pi24mmFilmId,
                     int *piDmin_R,
                     int *piDmin_G,
                     int *piDmin_B);
```

Parameters

iStripIndex

[in] The index of the strip for which information will be retrieved. Valid values are 0 to the number of strips – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piRollIndex

[out] The index of the roll to which this strip belongs. Possible values are 0 to the number of rolls – 1.

pbIsStrip

[out] TRUE if this strip was scanned in strip mode (multiple strips per roll).

piFilmColor

[out] The color of the film. See the FILM_COLOR_000 enumeration for possible values.

piFilmFormat

[out] The format of the film. See the FILM_FORMAT_000 enumeration for possible values.

plHeightHR

[out] The height of the strip from the high-resolution buffer in pixels.

plLengthHR

[out] The usable length of the strip from the high-resolution buffer in pixels.

plHeightLR

[out] The height of the strip from the low-resolution buffer in pixels.

plLengthLR

[out] The usable length of the strip from the low-resolution buffer in pixels.

piScanWarnings

[out] A list of warnings from the original scan. This is a bitwise sum of warning codes from the SCAN_WARNINGS_000 enumeration.

piFilmProduct

[out] The film product class as determined from the DX codes. If DX is not read, this will be -1.

piFilmSpecifier

[out] The film specifier as determined from the DX codes. If DX is not read, this will be -1.

pi24mmFilmId

[out] The film ID as determined from MOF data. If MOF is not read, this will be 0.

piDmin_R

[out] The calculated value of Dmin for the red channel. Dmin is the base density of the scanned film.

piDmin_G

[out] The calculated value of Dmin for the green channel. Dmin is the base density of the scanned film.

piDmin_B

[out] The calculated value of Dmin for the blue channel. Dmin is the base density of the scanned film.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex
EC_InvalidMemberVariable

Remarks

This method cannot be called if a long save operation is in progress, an error will occur.

See Also

ScanPictures

3.5.21 InsertPicture

The ISavePictures::InsertPicture method inserts a new picture into TLA's list of pictures. The new picture could be one that was missed in the scan. Specify a framing rectangle to be used on the high-resolution buffer to get the new picture.

```
HRESULT InsertPicture(int iIndex,
                      int iStripIndex,
                      long lLeftHR,
                      long lTopHR,
                      long lRightHR,
                      long lBottomHR);
```

Parameters

iIndex

[in] The index of the insertion location. The new picture will be inserted before this index. Valid values are 0 to the number of pictures – 1, or INDEX_First or INDEX_InsertPictureAtEnd from the INDEX_000 enumeration.

iStripIndex

[in] The index of the strip to which a picture should be inserted. Valid values are 0 to the number of strips – 1. A picture cannot be inserted if there are no strips.

lLeftHR

[in] The left edge of the new picture frame in the high-resolution buffer in pixels. Valid values must be >= 0.

lTopHR

[in] The top edge of the new picture frame in the high-resolution buffer in pixels. Valid values must be >= 0.

lRightHR

[in] The right edge of the new picture frame in the high-resolution buffer in pixels. Valid values must be less than the usable length of film, and the resulting width must be >= 64.

lBottomHR

[in] The bottom edge of the new picture frame in the high-resolution buffer in pixels. Valid values must be less than the film height, and the resulting height must be >= 64.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned. If the film is APS, E_NOTIMPL will be returned.

Additional Error Codes

EC_WorkerThreadExists	EC_NoPicturesOrStrips	EC_InvalidIndex
EC_InvalidMemberVariable	EC_InvalidParameter	EC_MemoryNew
EC_NotAllowedWithAps		

Remarks

This method cannot be called if a long save operation is in progress, an error will occur. When a picture is inserted, the other pictures with higher indexes are shifted up one. That way, the picture indexes always range from 0 to the number of pictures in the save group – 1. Pictures cannot be inserted into APS film.

See Also

DeletePicture

3.5.22 MoveOldestRollToSaveGroup

The ISavePictures::MoveOldestRollToSaveGroup method transfers all the pictures in the roll that has been in the scan group the longest (roll index 0) to the save group.

```
HRESULT MoveOldestRollToSaveGroup();
```

Parameters

None.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_NoPicturesOrStrips EC_WorkerThreadExists

Remarks

There must be at least one roll in the scan group, otherwise an error will occur. This method cannot be called if a long save operation is in progress, an error will also occur. It can be called even if a long scan operation is in progress, no error will occur. When a roll is moved, the other rolls with higher indexes are shifted down one. That way, the roll indexes always range from 0 to the number of rolls in the scan group – 1.

See Also

ReleaseSaveGroup

3.5.23 PutPictureColorSettings

The ISavePictures::PutPictureColorSettings method sets color information about a picture.

```
HRESULT PutPictureColorSettings(int iIndex,
                                int iRed,
                                int iGreen,
                                int iBlue,
                                int iBrightness,
                                int iContrast,
                                int iSharpness);
```

Parameters

iIndex

[in] The index of the picture for which settings are to be set. Valid values are 0 to the number of pictures – 1, or any choice from the INDEX_000 enumeration except INDEX_InsertPictureAtEnd.

iRed

[in] The amount of red saturation for this picture. The valid range is –1000 to 1000.

iGreen

[in] The amount of green saturation for this picture. The valid range is –1000 to 1000.

iBlue

[in] The amount of blue saturation for this picture. The valid range is –1000 to 1000.

iBrightness

[in] The brightness setting for this picture. The valid range is –1000 to 1000.

iContrast

[in] The contrast setting for this picture. The valid range is –1000 to 1000.

iSharpness

[in] The sharpness of the picture. The valid range is -1000 to 1000.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

If *iIndex* indicates multiple pictures, then pictures marked as hidden or not selected will be skipped.

The current values can be found by calling GetPictureColorSettings. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

GetPictureColorSettings PutPictureColorSettingsDifferential

3.5.24 PutPictureColorSettingsDifferential

The ISavePictures::PutPictureColorSettingsDifferential method changes color information about a picture differentially. For example, if the red setting of a picture is currently 280 and this method is called with *iRed* = 10, then the resulting red setting will be 290.

```
HRESULT PutPictureColorSettingsDifferential(int iIndex,
                                            int iRed,
                                            int iGreen,
                                            int iBlue,
                                            int iBrightness,
                                            int iContrast,
                                            int iSharpness);
```

Parameters

iIndex

[in] The index of the picture for which settings are to be changed. Valid values are 0 to the number of pictures – 1, or any choice from the INDEX_000 enumeration except INDEX_InsertPictureAtEnd.

iRed

[in] The amount of red saturation change for this picture. The resulting setting must be in the range –1000 to 1000.

iGreen

[in] The amount of green saturation change for this picture. The resulting setting must be in the range –1000 to 1000.

iBlue

[in] The amount of blue saturation change for this picture. The resulting setting must be in the range –1000 to 1000.

iBrightness

[in] The brightness setting change for this picture. The resulting setting must be in the range –1000 to 1000.

iContrast

[in] The contrast setting change for this picture. The resulting setting must be in the range –1000 to 1000.

iSharpness

[in] The sharpness setting change for this picture. The resulting setting must be in the range –1000 to 1000.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

If *iIndex* indicates multiple pictures, then pictures marked as hidden (INDEX_All) or not selected (INDEX_AllSelected) will be skipped.

The current values can be found by calling GetPictureColorSettings. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

GetPictureColorSettings PutPictureColorSettings

3.5.25 PutPictureFramingUserInfo

The ISavePictures::PutPictureFramingUserInfo method sets information about the user's framing parameters of a certain picture in the save group.

```
HRESULT PutPictureFramingUserInfo(int iIndex,
                                  long lLeftHR,
                                  long lTopHR,
                                  long lRightHR,
                                  long lBottomHR);
```

Parameters

iIndex

[in] The index of the picture for which to change user framing information. Valid values are 0 to the number of pictures in the save group – 1, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

lLeftHR

[in] The left edge of this picture's frame in the high-resolution buffer in pixels. The valid range is zero to 16 less than the length of the buffer.

lTopHR

[in] The top edge of this picture's frame in the high-resolution buffer in pixels. The valid range is zero to 16 less than the height of the buffer.

lRightHR

[in] The right edge of this picture's frame in the high-resolution buffer in pixels. The valid range is 16 to one less than the length of the buffer.

lBottomHR

[in] The bottom edge of this picture's frame in the high-resolution buffer in pixels. The valid range is 16 to one less than the height of the buffer.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

Use this function to adjust the position of the framing window or to apply cropping to the picture. This method cannot be called if a long save operation is in progress, an error will occur. The resulting frame must be at least 16 x 16 pixels. Call GetPictureFramingUserInfo to get the current user parameters or GetPictureFramingInfo to get the original framing parameters.

See Also

GetPictureFramingUserInfo GetPictureFramingInfo

3.5.26 PutPictureInfo

The ISavePictures::PutPictureInfo method changes information about a certain picture in the save group. Use GetPictureInfo to get the current values.

```
HRESULT PutPictureInfo(int iIndex,
                       int iFrameNumber,
                       BSTR bstrFileName,
                       BSTR bstrDirectory,
                       int iRotation,
                       int iSelectedHidden);
```

Parameters

iIndex

[in] The index of the picture for which information is to be changed. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

iFrameNumber

[in] The number of the frame. Valid values range from –6 to 1999. Negative values cause the frame name to take on the values shown in the table below, in Remarks. NOTE: If APS film was scanned with an F235C and magnetic data was read, then the frame numbers cannot be changed.

bstrFileName

[in] The name of the file where this picture will be saved with SaveToDisk. The filename should not have an extension. This parameter is used ONLY for the SaveToDisk method.

bstrDirectory

[in] The location of the file where this picture will be saved with SaveToDisk. This parameter is used ONLY for the SaveToDisk method.

iRotation

[in] The rotation of the picture. Valid values can be found in the ROTATE_000 enumeration.

iSelectedHidden

[in] Indicates whether the picture is hidden or selected. Valid values can be found in the S_OR_H_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex
EC_InvalidParameter EC_ChangingFrameNumberWithAps

Remarks

Use GetPictureInfo to get the current values. This method cannot be called if a long save operation is in progress, an error will occur. Setting the frame number to a negative value affects the frame name as follows:

Frame number	Frame name
-6	XX
-5	XXA
-4	X
-3	XA
-2	00
-1	00A

If this method is used to change the frame number, the frame name will change also. The client can call GetPictureInfo to get the new frame name. The frame numbers of APS film of which magnetic data was read cannot be changed.

See Also

GetPictureInfo SaveToDisk

3.5.27 PutPictureInfo1

The ISavePictures::PutPictureInfo1 method changes information about a certain picture in the save group. Use GetPictureInfo1 to get the current values.

```
HRESULT PutPictureInfo1(int iIndex,
                        int iFrameNumber,
                        BSTR bstrFileName,
                        BSTR bstrDirectory);
```

Parameters

iIndex

[in] The index of the picture for which information is to be changed. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

iFrameNumber

[in] The number of the frame. Valid values range from –6 to 1999. Negative values cause the frame name to take on the values shown in the Remarks for PutPictureInfo. NOTE: If APS film was scanned with an F235C and magnetic data was read, then the frame numbers cannot be changed.

bstrFileName

[in] The name of the file where this picture will be saved with SaveToDisk. The filename should not have an extension. This parameter is used ONLY for the SaveToDisk method.

bstrDirectory

[in] The location of the file where this picture will be saved with SaveToDisk. This parameter is used ONLY for the SaveToDisk method.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex
EC_InvalidParameter EC_ChangingFrameNumberWithAps

Remarks

Use GetPictureInfo1 to get the current values. This method cannot be called if a long save operation is in progress, an error will occur. This method is identical to PutPictureInfo with a few less parameters. The frame numbers of APS film of which magnetic data was read cannot be changed.

See Also

GetPictureInfo1 SaveToDisk PutPictureInfo

3.5.28 PutPictureLightingDirection

The ISavePictures::PutPictureLightingDirection method sets information about the direction of light of a certain picture in the save group. Lighting direction is NOT YET IMPLEMENTED.

```
HRESULT PutPictureLightingDirection(int iIndex,
                                    int iPictureLightingDirection);
```

Parameters

iIndex

[in] The index of the picture for which to set lighting information. Valid values are zero to one less than the number of pictures in the save group, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

iPictureLightingDirection

[out] The direction of light for this picture. Valid values can be found in the PICTURE_LIGHTING_DIRECTION_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

Remarks

The lighting direction can be retrieved with GetPictureLightingDirection and set with this function, but it is not yet implemented in the color correction algorithms.

See Also

GetPictureLightingDirection

3.5.29 PutPictureRedEyeSettings

The ISavePictures::PutPictureRedEyeSettings method retrieves information about the red-eye parameters of a certain picture in the save group. Red eye is NOT YET IMPLEMENTED.

```
HRESULT PutPictureRedEyeSettings(int iIndex,
                                 int iRedEye,
                                 long lLeftHR,
                                 long lTopHR,
                                 long lRightHR,
                                 long lBottomHR);
```

Parameters

iIndex

[in] The index of the picture for which to retrieve framing information. Valid values are zero to one less than the number of pictures in the save group, or `INDEX_Current` or `INDEX_First` from the INDEX_000 enumeration.

iRedEye

[out] The red-eye setting for this picture. See the RED_EYE_000 enumeration for possible values.

lLeftHR

[out] The left edge of this picture's red-eye rectangle in the high-res buffer in pixels.

lTopHR

[out] The top edge of this picture's red-eye rectangle in the high-res buffer in pixels.

lRightHR

[out] The right edge of this picture's red-eye rectangle in the high-res buffer in pixels.

lBottomHR

[out] The bottom edge of this picture's red-eye rectangle in the high-res buffer in pixels.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

Remarks

The red eye settings can be retrieved with the GetPictureRedEyeSettings function and set with this function, but red eye reduction is not yet implemented.

See Also

GetPictureRedEyeSettings

3.5.30 PutPictureRotation

The ISavePictures::PutPictureRotation method rotates a certain picture in the save group. Use GetPictureRotation to get the current value.

```
HRESULT PutPictureRotation(int iIndex,
                           int iRotation);
```

Parameters

iIndex

[in] The index of the picture for which information is to be changed. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piRotation

[in] The rotation of the picture. Valid values can be found in the ROTATE_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

Use GetPictureRotation to get the current value. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

GetPictureRotation

3.5.31 PutPictureSelection

The ISavePictures::PutPictureSelection method changes the selection/hidden parameter of a certain picture in the save group. Use GetPictureSelection to get the current value.

```
HRESULT PutPictureSelection(int iIndex,
                            int iSelectedHidden);
```

Parameters

iIndex

[in] The index of the picture for which information is to be changed. Valid values are 0 to the number of pictures – 1, or INDEX_Current or INDEX_First from the INDEX_000 enumeration.

piSelectedHidden

[in] Indicates whether the picture is hidden or selected. Valid values can be found in the S_OR_H_000 enumeration.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex
EC_InvalidParameter

Remarks

Use GetPictureSelection to get the current values. This method cannot be called if a long save operation is in progress, an error will occur.

See Also

GetPictureSelection

3.5.32 PutSaveInfo

The ISavePictures::PutSaveInfo method changes information about how pictures will be saved with the SaveToDisk method. Use GetSaveInfo to get the current values.

```
HRESULT PutSaveInfo(int iDefaultFilmProduct,
                    int iDefaultFilmSpecifier,
                    BSTR bstrDefaultDirectory);
```

Parameters

iDefaultFilmProduct
[in] The default film product class. This parameter is not used.

iDefaultFilmSpecifier
[in] The default film specifier. This parameter is not used.

bstrDefaultDirectory
[in] The default directory. The initial value is "c:\temp".

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists EC_NoPicturesOrStrips EC_InvalidIndex

Remarks

Use GetSaveInfo to get the current values. This method cannot be called if a long save operation is in progress, an error will occur. Changing the default directory with this function will affect pictures that are scanned after the call, but it will not change the default directory for pictures already in the scan group. If the default directory of a picture already in the scan group needs to be changed, move it to the save group and then call either PutPictureInfo or PutPictureInfo1.

See Also

GetSaveInfo SaveToDisk PutPictureInfo PutPictureInfo1

3.5.33 ReleaseSaveGroup

The ISavePictures::ReleaseSaveGroup method deletes all the pictures (and all the buffers) in the save group. These pictures cannot be recovered.

```
HRESULT ReleaseSaveGroup();
```

Parameters

None.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WorkerThreadExists

Remarks

This method cannot be called if a long save operation is in progress, an error will occur.

See Also

3.5.34 SaveCancel

The ISavePictures::SaveCancel method will stop any long save operation currently running.

```
HRESULT SaveCancel();
```

Parameters

None.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_WIN_SetEvent

Remarks

Calling this function will stop any long save operation currently running, but it won't reverse the progress of the operation. For example, if saving, this function will stop the save, but any images already saved will remain in memory or on the disk.

See Also

SaveToClientMemory SaveToDisk

3.5.35 SaveToClientMemory

The ISavePictures::SaveToClientMemory method saves one or more pictures to memory the client has established with a call to ClientMemoryBufferAdd.

```
HRESULT SaveToClientMemory(int iIndex,
                           int iSaveControl,
                           int iBoundingWidth,
                           int iBoundingHeight,
                           int iRotation,
                           int iScalingMethod,
                           int iFileFormatSaveToMemory,
                           BOOL bUseWorkerThread);
```

Parameters

iIndex

[in] The index(es) of the picture(s) to be saved. Valid values are 0 to the number of pictures in the save group - 1, and any value from the INDEX_000 enumeration (except INDEX_InsertPictureAtEnd).

iSaveControl

[in] A bitwise sum of codes from the SAVE_CONTROL_000 enumeration. This indicates the options that are desired for this save operation.

iBoundingWidth

[in] The width of the resulting picture in pixels (the picture may be scaled up or down). If *iSaveControl* includes SAV_DoNotScaleUp, the picture will not be scaled up. If *iSaveControl* includes SAV_SizeLimitForDisplay or SAV_SizeLimitForSave, then the bounding width must be at least 16 pixels.

iBoundingHeight

[in] The height of the resulting picture in pixels (the picture may be scaled up or down). If *iSaveControl* includes SAV_DoNotScaleUp, the picture will not be scaled up. If *iSaveControl* includes SAV_SizeLimitForDisplay or SAV_SizeLimitForSave, then the bounding height must be at least 16 pixels.

iRotation

[in] Indicates how the picture should be rotated. See the ROTATE_000 enumeration for valid values. The image can also be mirrored by including MIRROR_L_R with a rotation value in a bitwise OR operation. If *iSaveControl* includes the value SAV_UseCurrentRotation, then this parameter is ignored.

iScalingMethod

[in] Indicates the scaling method desired (whether to scale is indicated with the *iBoundingWidth* and *iBoundingHeight* parameters). See the SCALING_METHOD_000 enumeration for valid values.

iFileFormatSaveToMemory

[in] Indicates the format of the file to save. Valid values can be found in the FILE_FORMAT_SAVE_TO_MEMORY_000 enumeration.

bUseWorkerThread

[in] TRUE if a separate processing thread should be used for this operation. If *iIndex* indicates more than one picture, then this parameter must be TRUE.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_ClientMemoryBufferInUse EC_ImageNotPlanar EC_InvalidIndex
EC_InsufficientMemoryForSaveToMemory EC_InvalidMemberVariable
EC_InvalidParameter EC_MemoryNew EC_NoClientMemoryBuffer
EC_NoPicturesOrStrips EC_NoWorkerThreadForMultipleSaveToMemory
EC_QueryInterface EC_TimeOut EC_UnableToCreateWorkerThread
EC_WorkerThreadClientSignal EC_WorkerThreadExists EC_WrongByteCount
EC_WorkerThreadStartTimeout

ERROR_CODES_020 ERROR_CODES_040

EC_PFS_FilePointerDeleted EC_PFS_InvalidPointer EC_PFS_NullFilePointer
EC_PFS_PartitionSelected EC_PFS_ReadPastEOF

EC_WIN_FileRead EC_WIN_ResetEvent EC_WIN_SetFilePointerEx
EC_WIN_WaitForSingleObject

Remarks

If *iIndex* indicates multiple pictures, then pictures marked as hidden or not selected will be skipped.

The pictures are saved to the oldest client memory buffer (the buffer which was added first, index 0). Once saved, this function will remove the used buffer from TLA's list of buffers (the buffer will still exist, just not in TLA's list).

This may be a long save operation, depending on the value of *bUseWorkerThread*. If it is a long operation, this method cannot be called if another long save operation is in progress, an error will occur. Once started, the client will get progress messages through the callback interface until complete. However, the client can stop the save prematurely by calling SaveCancel.

Callbacks

Operation	Status	Comment
WTO_SaveProgress	WTP_Initialize	Always, right after WT startup
WTO_SaveProgress	WTP_ProgressStart	Always
WTO_SaveProgress	WTP_ProgressStart + x	< WTP_ProgressEnd
WTO_SaveProgress	WTP_ProgressComplete	If successful
WTO_SaveError	WTP_ProgressComplete	If error

See Also

ClientMemoryBufferAdd Awake SaveCancel

3.5.36 SaveToDisk

The ISavePictures::SaveToDisk method saves one or more pictures to disk.

```
HRESULT SaveToDisk(int iIndex,
                   BSTR bstrFileName,
                   int iSaveControl,
                   int iBoundingWidth,
                   int iBoundingHeight,
                   int iRotation,
                   int iScalingMethod,
                   int iFileFormat,
                   int iCompression,
                   int iDpi,
                   int iColorBits);
```

Parameters

iIndex

[in] The index(es) of the picture(s) to be saved. Valid values are zero to one less than the number of pictures in the save group, and any value from the INDEX_000 enumeration (except INDEX_InsertPictureAtEnd).

bstrFileName

[in] The name of the file to save to (the filename should not have an extension), or NULL. See Remarks.

iSaveControl

[in] A bitwise sum of codes from the SAVE_CONTROL_000 enumeration. This indicates the options that are desired for this save operation.

iBoundingWidth

[in] The width of the resulting picture in pixels (the picture may be scaled up or down). If *iSaveControl* includes SAV_DoNotScaleUp, the picture will not be scaled up. If *iSaveControl* includes SAV_SizeLimitForDisplay or SAV_SizeLimitForSave, then the bounding width must be at least 16.

iBoundingHeight

[in] The height of the resulting picture in pixels (the picture may be scaled up or down). If *iSaveControl* includes SAV_DoNotScaleUp, the picture will not be scaled up. If *iSaveControl* includes SAV_SizeLimitForDisplay or SAV_SizeLimitForSave, then the bounding height must be at least 16.

iRotation

[in] Indicates how the picture should be rotated. See the ROTATE_000 enumeration for valid values. The image can also be mirrored by including MIRROR_L_R with a rotation value in a bitwise OR operation. If *iSaveControl* includes the value SAV_UseCurrentRotation, then this parameter is ignored.

iScalingMethod

[in] Indicates the scaling method desired (whether to scale is indicated with the *iBoundingWidth* and *iBoundingHeight* parameters). See the SCALING_METHOD_000 enumeration for valid values.

iFileFormat

[in] Indicates the format of the file to save. Valid values can be found in the FILE_FORMAT_000 enumeration.

iCompression

[in] The amount of file compression to use for JPEG files. The valid range is 0 to 100 (0 is maximum compression, 100 is minimum compression).

iDpi

[in] This parameter is stored in the header of the file. However, if the file type is BMP, then this parameter is ignored and the dpi in the header will be set to 72. There is no limit to the valid range of this parameter.

iColorBits

[in] This parameter is not yet implemented.

Return Value

If successful, S_OK is returned, otherwise an error code of E_FAIL will be returned.

Additional Error Codes

EC_CreateDirectoryInvalidForm EC_InvalidIndex EC_InvalidParameter
EC_InvalidMemberVariable EC_MemoryNew EC_NoPicturesOrStrips
EC_OneFileNameForMultipleSaves EC_UnableToCreateWorkerThread
EC_WorkerThreadClientSignal EC_WorkerThreadExists
EC_WorkerThreadStartTimeout EC_WrongByteCount EC_TimeOut

ERROR_CODES_020 ERROR_CODES_040

EC_WIN_FileRead EC_WIN_ResetEvent EC_WIN_SetFilePointerEx
EC_WIN_WaitForSingleObject

Remarks

If *iIndex* indicates multiple pictures, then *bstrFileName* must be NULL. Also, pictures marked as hidden or not-selected may be skipped, depending on the value of *iIndex*.

The pictures are saved to the default directory with the filename specified. The default directory is “c:\temp” but can be changed with a call to PutSaveInfo. If this directory doesn’t exist, it will be created.

The filename is specified with the input parameter, or if *bstrFileName* is NULL, then the default filename for each picture is used. The default is the frame name (e.g. "4"), but can be changed with a call to PutPictureInfo or PutPictureInfo1. The filename should not have an extension (it will be determined from the file format).

This is a long save operation. This method cannot be called if another long save operation is in progress, an error will occur. Once started, the client will get progress messages through the callback interface until complete. However, the client can stop the save prematurely by calling SaveCancel.

Callbacks

Operation	Status	Comment
WTO_SaveProgress	WTP_Initialize	Always, right after WT startup
WTO_SaveProgress	WTP_ProgressStart	Always
WTO_SaveProgress	WTP_ProgressStart + x	< WTP_ProgressEnd
WTO_SaveProgress	WTP_ProgressComplete	If successful
WTO_SaveError	WTP_ProgressComplete	If error

See Also

PutPictureInfo PutPictureInfo1 PutSaveInfo Awake
SaveCancel

4 Enumerations

Listed here are the enumerations available from the TLA.idl file. Not all enumerations are listed, specifically, the CLASS_NAMES_000, the FUNCTION_NAMES_000, and the DRIVER_FUNCTION_000 enumerations. Please consult the TLA.idl file for these enumerations. Also, the error code enumerations are listed in the next section.

4.1 WORKER_THREAD_OPERATION_000

These enumerations are used to report progress or errors by way of the client callback interface - Awake(*lOperation*, *lStatus*). These values will be used for the *lOperation* parameter. The value of *lStatus* will come from the WORKER_THREAD_PROGRESS_000 or HARDWARE_CB_000 enumerations.

- **WTO_InitializeProgress**
  Used to report the progress of some initialization process.

- **WTO_InitializeError**
  Used to report an error during initialization.

- **WTO_FirmwareUpdateApsProgress**
  Used to report the progress of an automatic APS firmware update.

- **WTO_FirmwareUpdateApsError**
  Used to report an error during an automatic APS firmware update.

- **WTO_FirmwareUpdateCcdProgress**
  Used to report the progress of an automatic CCD firmware update.

- **WTO_FirmwareUpdateCcdError**
  Used to report an error during an automatic CCD firmware update.

- **WTO_FirmwareUpdateDxProgress**
  Used to report the progress of an automatic DX firmware update.

- **WTO_FirmwareUpdateDxError**
  Used to report an error during an automatic DX firmware update.

- **WTO_FirmwareUpdateLampProgress**
  Used to report the progress of an automatic lamp firmware update.

- **WTO_FirmwareUpdateLampError**
  Used to report an error during an automatic lamp firmware update.

- **WTO_FirmwareUpdateMotorProgress**
  Used to report the progress of an automatic motor firmware update.

- **WTO_FirmwareUpdateMotorError**
  Used to report an error during an automatic motor firmware update.

- **WTO_HardwareProgress**
  Used to report that film was sensed at the entry or exit. See HARDWARE_CB_000.

- **WTO_HardwareError**
  Used to report some hardware error or warning. See HARDWARE_CB_000.

- **WTO_HardwareAPSProgress**
  Used to report some informational message concerning APS (e.g. a cartridge was loaded). See HARDWARE_CB_APS_000.

- **WTO_HardwareAPSError**
  Used to report some APS hardware error or warning. See HARDWARE_CB_APS_000.

- **WTO_FilmTrackTestProgress**
  Used to report the progress of a film track test.

- **WTO_FilmTrackTestError**
  Used to report an error during a film track test. *lStatus* will be WTP_ProgressComplete. Call FilmTrackTestResults to get the error codes.

- **WTO_DiagnosticsProgress**
  Used to report the progress of a diagnostics test.

- **WTO_DiagnosticsError**
  Used to report an error during a diagnostics test.

- **WTO_CorrectionsProgress**
  Used to report the progress of some force-corrections call.

- **WTO_CorrectionsError**
  Used to report an error during a force-corrections call.

- **WTO_ExerciseSteppersProgress**
  Used to report the progress of exercising the stepper motors during initialization.

- **WTO_ExerciseSteppersError**
  Used to report an error while exercising the stepper motors during initialization.

- **WTO_LampWarmUpProgress**
  Used to report the progress of the lamp warm-up.

- **WTO_LampWarmUpError**
  Used to report an error during lamp warm-up.

- **WTO_AdvanceFilmProgress**
  Used to report the progress of a film advance.

- **WTO_AdvanceFilmError**
  Used to report an error during a film advance.

- **WTO_PutFilmGuidePositionProgress**
  Used to report the progress of a film guide positioning operation.

- **WTO_PutFilmGuidePositionError**
  Used to report an error during a film guide positioning operation.

- **WTO_PutFilmPressureRollersPositionProgress**
  Used to report the progress of a pressure roller positioning operation.

- **WTO_PutFilmPressureRollersPositionError**
  Used to report an error during a pressure roller positioning operation.

- **WTO_F235C_ManualRetractProgress**
  Used to report the progress of an APS manual retract operation.

- **WTO_F235C_ManualRetractError**
  Used to report an error during an APS manual retract operation.

- **WTO_ScanProgress**
  Used to report the progress of a scan.

- **WTO_ScanError**
  Used to report an error during a scan.

- **WTO_ImportFromFileProgress**
  Used to report the progress of an import from file operation.

- **WTO_ImportFromFileError**
  Used to report an error during an import from file operation.

- **WTO_SaveProgress**
  Used to report the progress of a save operation.

- **WTO_SaveError**
  Used to report an error during a save operation.

- **WTO_Last**
  Indicates the last worker-thread-operation enumeration (equals WTO_SaveError).

4.2 WORKER_THREAD_PROGRESS_000

These enumerations are used to report progress or errors by way of the client callback interface - Awake(*lOperation*, *lStatus*). These values will be used for the *lStatus* parameter. The *lOperation* parameter will be nearly any one of the WORKER_THREAD_OPERATION_000 enumerations.

- **WTP_Initialize**
  Indicates that the operation has initialized.

- **WTP_ProgressStart**
  Indicates that the operation has started.

- **WTP_ProgressEnd**
  Indicates that the operation has finished.

- **WTP_ProgressComplete**
  Indicates that the operation has completed.

4.3 HARDWARE_CB_000

This enumeration is used to report hardware issues by way of the client callback interface - Awake(*lOperation*, *lStatus*). These values will be used for the *lStatus* parameter. The *lOperation* parameter will be either WTO_HardwareProgress or WTO_HardwareError.

If the client receives a WTO_HardwareError and the status indicates a warning condition (HARDWARE_CB_xxx_WARNING), then scanning can continue, but a service technician should be called to troubleshoot the error. If the lamp's heat sink temperature is too high, the operator should first clean the air filter.

If the client receives a WTO_HardwareError and the status indicates an error condition (HARDWARE_CB_xxx_FAULT or HARDWARE_CB_xxx_ERROR), then a service technician is required to troubleshoot and diagnose the error before scanning can continue. The exceptions are receiving a lamp burnout error (which the operator can repair) or a lamp temperature high error (the operator should clean the air filter). If the error occurred during a scan, the operator may try to salvage the scan by processing the order normally to see how many pictures from the roll were captured prior to the error.

The two cases where the client will get a WTO_HardwareProgress are when the *lStatus* parameter is HARDWARE_CB_FILM_SENSE_ENTRY or HARDWARE_CB_FILM_SENSE_EXIT.

- **HARDWARE_CB_HOST_FAULT**
  Indicates a polling error with the F235 host controller.

- **HARDWARE_CB_DX_FAULT**
  Indicates a polling error with the DX controller.

- **HARDWARE_CB_LAMP_FAULT**
  Indicates a polling error with the lamp controller.

- **HARDWARE_CB_CCD_FAULT**
  Indicates a polling error with the CCD controller.

- **HARDWARE_CB_MOTOR_FAULT**
  Indicates a polling error with the motor controller.

- **HARDWARE_CB_LAMP_CURRENT_HI_WARNING**
  Warns the client that the lamp board's current draw is high. Replace the bulb.

- **HARDWARE_CB_LAMP_CURRENT_HI_ERROR**
  Warns the client that the lamp board's current draw is too high to continue.

- **HARDWARE_CB_LAMP_TEMP_HI_WARNING**
  Warns the client that the lamp's heat sink temperature is high. Clean the air filter.

- **HARDWARE_CB_LAMP_TEMP_HI_ERROR**
  Warns the client that the lamp's heat sink temperature is too high to continue. Turn off the scanner and clean the air filter.

- **HARDWARE_CB_LAMP_BURN_OUT_ERROR**
  Informs the client that the lamp has burned out. The operator will need to replace the lamp.

- **HARDWARE_CB_LAMP_FAN_SLOW_WARNING**
  Warns the client that the lamp's fan is running somewhat slow.

- **HARDWARE_CB_LAMP_FAN_SLOW_ERROR**
  Informs the client that the lamp's fan is running very slow and the lamp may overheat.

- **HARDWARE_CB_MOTOR_STEPPER_CCD**
  Used to report a CCD stepper motor positioning error.

- **HARDWARE_CB_MOTOR_STEPPER_LENS**
  Used to report a lens stepper motor positioning error.

- **HARDWARE_CB_MOTOR_FILTER_WHEEL**
  Used to report a filter wheel stepper motor positioning error.

- **HARDWARE_CB_MOTOR_FILM_GUIDE**
  Used to report a film guide stepper motor positioning error.

- **HARDWARE_CB_CLEANING_REQUIRED**
  Warns the client that a fixed pattern correction detected an obstruction and the scanner should be cleaned.

- **HARDWARE_CB_FILM_EMULSION_DOWN**
  Warns the client that the film was inserted emulsion side down.

- **HARDWARE_CB_FILM_TAIL_FIRST**
  Warns the client that the film was inserted tail first.

- **HARDWARE_CB_FILM_SENSE_ENTRY**
  Informs the client that the film was sensed at the entry point. If scanning APS film with the autoloader in an F235C scanner, this callback is not used.

- **HARDWARE_CB_FILM_SENSE_EXIT**
  Informs the client that the film was sensed at the exit point.

- **HARDWARE_CB_INFO_BITS**
  Indicates some kind of informational message (film was sensed at the entry or exit).

- **HARDWARE_CB_WARNING_BITS**
  Indicates some kind of warning message (lamp current is high, temperature is high, cleaning is required, or the film was inserted emulsion-side down or tail first).

- **HARDWARE_CB_ERROR_BITS**
  Indicates some kind of error message (anything besides a warning or info message).

4.4 HARDWARE_CB_APS_000

This enumeration is used to report APS hardware issues by way of the client callback interface - Awake(*lOperation*, *lStatus*). These values will be used for the *lStatus* parameter. The *lOperation* parameter will be either WTO_HardwareAPSProgress or WTO_HardwareAPSError.

If the client receives a WTO_HardwareAPSError and the status indicates a warning condition (HARDWARE_CB_APS_xxx_WARNING), then scanning can continue, but the client should resolve the problem as soon as possible.

If the client receives a WTO_HardwareAPSError and the status indicates an error condition (HARDWARE_CB_APS_xxx_FAULT or HARDWARE_CB_APS_xxx_ERROR), then the client is required to troubleshoot and diagnose the error before scanning can continue. If the error occurred during a scan, the operator may try to salvage the scan by processing the order normally to see how many pictures from the roll were captured prior to the error.

The one case where the client will get a WTO_HardwareProgress is when the *lStatus* parameter is HARDWARE_CB_APS_CARTRIDGE_LOADED.

- **HARDWARE_CB_APS_FAULT**
  Indicates a polling error with the APS (MOF) controller.

- **HARDWARE_CB_APS_CARTRIDGE_LOADED**
  Informs the client that an APS cartridge was loaded (informational).

- **HARDWARE_CB_APS_FILM_JAM_EXTRACT**
  Indicates a problem extracting film from the APS cartridge, probably due to a film jam.

- **HARDWARE_CB_APS_FILM_JAM_SCAN**
  Indicates a problem scanning film from the APS cartridge, probably due to a film jam.

- **HARDWARE_CB_APS_FILM_JAM_RETRACT**
  Indicates a problem retracting film into the APS cartridge, probably due to a film jam.

- **HARDWARE_CB_APS_EJECT_ERROR**
  Indicates that the APS eject button was pressed when film was out of the cartridge (e.g. during a scan or during retraction).

- **HARDWARE_CB_APS_UNPROCESSED_ERROR**
  Indicates that the APS film has not been processed and cannot be scanned.

- **HARDWARE_CB_APS_CART_UNPACKED_ERROR**
  Indicates that an APS cartridge with extracted film was found in the scanner when it was powered up. The film may be jammed.

- **HARDWARE_CB_APS_PARK_INIT_ERROR**
  Indicates that the APS park sensor could not be initialized and may have to be replaced.

- **HARDWARE_CB_APS_PARK_ERROR**
  Indicates that the APS cartridge could not be properly parked after retraction (the film's status indicator could not be returned to the "processed" position, number 4).

- **HARDWARE_CB_APS_INFO_BITS**
  Indicates some kind of informational message (the APS cartridge was loaded).

- **HARDWARE_CB_APS_FILM_JAM_BITS**
  Indicates some kind of warning message (the film is jammed).

- **HARDWARE_CB_APS_ERROR_BITS**
  Indicates some kind of error message (anything besides a warning or info message).

4.5 SCANNER_TYPE_000

These values indicate the type of scanner and are used in the GetScannerInfo000 function.

- **SCANNER_TYPE_F_235**
  Specifies the F235 or F235*plus* scanner.

- **SCANNER_TYPE_F_235C**
  Specifies the F235C scanner (APS cartridge loader and MOF reader built-in).

4.6 SCANNER_VERSION_HW_000

These values indicate the version of scanner hardware and are used in the GetScannerInfo000 function.

- **SCANNER_VERSION_HW_PRODUCTION**
  Indicates an F235 scanner.

- **SCANNER_VERSION_HW_BRIDGE**
  Reserved for future use.

- **SCANNER_VERSION_HW_FLATBELT**
  Indicates an F235*plus* or F235C scanner.
- **SCANNER_VERSION_HW_RFT_SPLICE**
  Indicates an F235plus that is able to transport splice film.

4.7 FILM_FORMAT_000

This enumeration is used to indicate the format of the film in the scanner. These values are used in the functions ForceCorrections, GetFilmGuidePosition, GetScannerInfoPreFrame, GetScannerInfoPreFrameUser, PutFilmGuidePosition, PutScannerInfoPreFrameUser, ScanPictures, and GetStripInfo.

- **FILM_FORMAT_INDETERMINATE**
  Indicates that the film format could not be determined. The client should never see or have to use this.

- **FILM_FORMAT_24MM**
  Indicates the film is 24mm (APS).

- **FILM_FORMAT_35MM**
  Indicates the film is 35mm.

- **FILM_FORMAT_70MM**
  Not applicable to the F235.

- **FILM_FORMAT_MOUNTED**
  Not applicable to the F235.

- **FILM_FORMAT_IMPORTED**
  Indicates the film was imported from file.

4.8 STRIP_MODE_000

This enumeration is used to indicate the strip mode to use. These values are used in the ScanPictures function.

- **STRIP_MODE_FULL_ROLL**
  Indicates a full roll is to be scanned (up to 40 frames). The scanner will stop at the end of film. This is the only valid choice when scanning 24mm film.

- **STRIP_MODE_MULTI_STRIPS_4_FRAMES**
  Indicates that multiple strips of 4 frames each are to be scanned. It is the operator's responsibility to insure a two-inch gap (or more) between strips. Once all strips have been scanned, the scanner can be stopped with a call to ScanCancel or just wait for a film timeout. TLA will then combine the strips into one roll.

- **STRIP_MODE_MULTI_STRIPS_5_FRAMES**
  Indicates that multiple strips of 5 frames each are to be scanned. It is the operator's responsibility to insure a two-inch gap (or more) between strips. Once all strips have been scanned, the scanner can be stopped with a call to ScanCancel or just wait for a film timeout. TLA will then combine the strips into one roll.

- **STRIP_MODE_MULTI_STRIPS_6_FRAMES**
  Indicates that multiple strips of 6 frames each are to be scanned. It is the operator's responsibility to insure a two-inch gap (or more) between strips. Once all strips have been scanned, the scanner can be stopped with a call to ScanCancel or just wait for a film timeout. TLA will then combine the strips into one roll.

4.9 FRAME_SIZES_000

These values are used to indicate the height or width of the frame in the low or high resolution buffer. This will depend on the scan resolution and film format. They are used in the functions GetScannerInfoPreFrame, GetScannerInfoPreFrameUser and PutScannerInfoPreFrameUser.

- **FRAME_SIZES_HR_HEIGHT_BASE4_24**
  Indicates the height of the frame in the high-resolution buffer for scanning 24mm film at base4 resolution.

- **FRAME_SIZES_HR_WIDTH_BASE4_24**
  Indicates the width of the frame in the high-resolution buffer for scanning 24mm film at base4 resolution.

- **FRAME_SIZES_HR_HEIGHT_BASE8_24**
  Indicates the height of the frame in the high-resolution buffer for scanning 24mm film at base8 resolution.

- **FRAME_SIZES_HR_WIDTH_BASE8_24**
  Indicates the width of the frame in the high-resolution buffer for scanning 24mm film at base8 resolution.

- **FRAME_SIZES_HR_HEIGHT_BASE16_24**
  Indicates the height of the frame in the high-resolution buffer for scanning 24mm film at base16 resolution.

- **FRAME_SIZES_HR_WIDTH_BASE16_24**
  Indicates the width of the frame in the high-resolution buffer for scanning 24mm film at base16 resolution.

- **FRAME_SIZES_HR_HEIGHT_BASE4_35**
  Indicates the height of the frame in the high-resolution buffer for scanning 35mm film at base4 resolution.

- **FRAME_SIZES_HR_WIDTH_BASE4_35**
  Indicates the width of the frame in the high-resolution buffer for scanning 35mm film at base4 resolution.

- **FRAME_SIZES_HR_HEIGHT_BASE8_35**
  Indicates the height of the frame in the high-resolution buffer for scanning 35mm film at base8 resolution.

- **FRAME_SIZES_HR_WIDTH_BASE8_35**
  Indicates the width of the frame in the high-resolution buffer for scanning 35mm film at base8 resolution.

- **FRAME_SIZES_HR_HEIGHT_BASE16_35**
  Indicates the height of the frame in the high-resolution buffer for scanning 35mm film at base16 resolution.

- **FRAME_SIZES_HR_WIDTH_BASE16_35**
  Indicates the width of the frame in the high-resolution buffer for scanning 35mm film at base16 resolution.

- **FRAME_SIZES_LR_HEIGHT_BASE4_24**
  Indicates the height of the frame in the low-resolution buffer for scanning 24mm film at base4 resolution.

- **FRAME_SIZES_LR_WIDTH_BASE4_24**
  Indicates the width of the frame in the low-resolution buffer for scanning 24mm film at base4 resolution.

- **FRAME_SIZES_LR_HEIGHT_BASE8_24**
  Indicates the height of the frame in the low-resolution buffer for scanning 24mm film at base8 resolution.

- **FRAME_SIZES_LR_WIDTH_BASE8_24**
  Indicates the width of the frame in the low-resolution buffer for scanning 24mm film at base8 resolution.

- **FRAME_SIZES_LR_HEIGHT_BASE16_24**
  Indicates the height of the frame in the low-resolution buffer for scanning 24mm film at base16 resolution.

- **FRAME_SIZES_LR_WIDTH_BASE16_24**
  Indicates the width of the frame in the low-resolution buffer for scanning 24mm film at base16 resolution.

- **FRAME_SIZES_LR_HEIGHT_BASE4_35**
  Indicates the height of the frame in the low-resolution buffer for scanning 35mm film at base4 resolution.

- **FRAME_SIZES_LR_WIDTH_BASE4_35**
  Indicates the width of the frame in the low-resolution buffer for scanning 35mm film at base4 resolution.

- **FRAME_SIZES_LR_HEIGHT_BASE8_35**
  Indicates the height of the frame in the low-resolution buffer for scanning 35mm film at base8 resolution.

- **FRAME_SIZES_LR_WIDTH_BASE8_35**
  Indicates the width of the frame in the low-resolution buffer for scanning 35mm film at base8 resolution.

- **FRAME_SIZES_LR_HEIGHT_BASE16_35**
  Indicates the height of the frame in the low-resolution buffer for scanning 35mm film at base16 resolution.

- **FRAME_SIZES_LR_WIDTH_BASE16_35**
  Indicates the width of the frame in the low-resolution buffer for scanning 35mm film at base16 resolution.

4.10 FILM_COLOR_000

These values indicate the color of film scanned or to be scanned. They are used in the functions ForceCorrections, LampManualControl, ScanPictures and GetStripInfo.

- **FILM_COLOR_NEGATIVE**
  Indicates the film is color negative.

- **FILM_COLOR_POSITIVE**
  Indicates the film is color positive.

- **FILM_COLOR_BnW_NORMAL**
  Indicates the film is normal black & white.

- **FILM_COLOR_BnW_C41**
  Indicates the film is C41 black & white.

- **FILM_COLOR_BnW_ANY**
  Indicates the film is either FILM_COLOR_BnW_NORMAL or FILM_COLOR_BnW_C41.

- **FILM_COLOR_LAMP_OFF**
  Used for LampManualControl only.

- **FILM_COLOR_LAMP_STANDBY**
  Used for LampManualControl only.

- **FILM_COLOR_FILTER_WHEEL_BLOCKED**
  Used for calibration only.

- **FILM_COLOR_IMPORTED_BnW**
  Indicates the film was imported from file and is black & white.

- **FILM_COLOR_IMPORTED_Color**
  Indicates the film was imported from file and is color.

- **FILM_COLOR_IMPORTED**
  Indicates the film is either FILM_COLOR_IMPORTED_BnW or FILM_COLOR_IMPORTED_Color.

4.11 FRAME_TYPE_000

This enumeration is used to indicate the aspect ratio of a frame for 24mm film. These values are used in the functions GetPictureInfo and GetPictureInfo2. The frame type affects the frame name.

- **PRINT_ASPECT_RATIO_H**
  Used for high vision framing. The frame name will have an “H” suffix.

- **PRINT_ASPECT_RATIO_P**
  Used for panoramic framing. The frame name will have a “P” suffix.

- **PRINT_ASPECT_RATIO_C**
  Used for classic framing. The frame name will have a “C” suffix.

4.12 RESOLUTION_000

These values indicate the scan resolution. They are used in the functions ForceCorrections, GetScannerInfoPreFrame, GetScannerInfoPreFrameUser, PutScannerInfoPreFrameUser, ResetFactoryDefaults and ScanPictures.

- **RESOLUTION_BASE_4**
  This is the lowest resolution. The number of pixels depends on the film format and the buffer resolution (high or low); for example, for 35mm film, the high resolution buffer will be 1000 x 1500 pixels.

- **RESOLUTION_BASE_8**
  This is medium resolution and provides twice as many pixels as base4. For example, for 35mm film, the high-resolution buffer will be 1400 x 2100 pixels.

- **RESOLUTION_BASE_16**
  This is the highest resolution and provides twice as many pixels as base8. For example, for 35mm film, the high-resolution buffer will be 2000 x 3000.

4.13 INITIALIZE_CONTROL_000

These values provide initialization options. They are used in the function InitializeScanner. Start with 0, then add in the options desired by using the bitwise sum operator (|).

- **INITIALIZE_ProgressUpdatesAsPercent**
  Indicates that progress updates (through the Awake function) should be reported as a percent. Otherwise, they will be reported as absolute values. The one exception is scan updates, which will always report as absolute values (the scanner doesn't know how many pictures or strips are going to be scanned).

- **INITIALIZE_FirmwareUpdate**
  Indicates that TLA should look for newer firmware and attempt to update the scanner. **NOTE**: It is **highly** recommended that clients include this option so that every scanner operates with the latest firmware. Without the latest firmware, a scanner may not operate properly.

- **INITIALIZE_MotorSelfTest**
  Indicates that TLA should exercise the stepper motors during initialization.

4.14 SCAN_CONTROL_000

These values are used to control how a scan is performed. They are used in the function ScanPictures. Start with 0, then add in the options desired by using the bitwise sum operator (|).

- **SCAN_Read_DX**
  Not used. The scanner will always try to read DX codes.

- **SCAN_AggressiveFraming**
  Indicates that the scanner should use aggressive framing.

- **SCAN_RFT_SenseSplice**
  Indicates the scan should stop when the CCD senses an opaque splice instead of stopping when the CCD senses open gate. The scanner must have a hardware version of SCANNER_VERSION_HW_RFT_SPLICE in order to use this option or an error will occur.

- **SCAN_UseScratchRemoval**
  Indicates that the scanner should use the Digital ICE™ scratch removal software. Scratch removal cannot be used with FILM_COLOR_BnW_NORMAL.

- **SCAN_Use24mmAutoLoader**
  Indicates that the scanner should use the APS auto loader. If this option is not specified and the film type is 24mm, then the scanner expects the operator to feed in the film. This option is valid only for the F235C scanner.

- **SCAN_Use24mmAutoLoaderMOF**
  Indicates that the scanner should attempt to read MOF data from the film. If this parameter is specified, the SCAN_Use24mmAutoLoader must also be specified, and the SCAN_Use24mmExternalFileMOF must not be specified.

- **SCAN_Use24mmAutoLoaderCalibrate**
  Reserved for future use.

- **SCAN_Use24mmExternalFileMOF**
  Reserved for future use.

- **SCAN_HasFilmDrag**
  Indicates that you expect extra drag on the film. This parameter is automatically used with APS film from a cartridge, but may be used if the film is coming from some sort of auto loader and you expect drag. (See the User's Guide for more explanation.)

- **SCAN_PreScan**
  Indicates that the scanner should perform a pre-scan. This will perform all the operations normally done prior to scanning (i.e. warming up the lamp, adjusting the film guide, etc.) without actually starting the scan.

- **SCAN_LampWarmUp**

Indicates that the scanner should warm up the lamp prior to scanning. It is highly recommened that this option always be included.

4.15 INITIALIZE_WARNINGS_000

These values are used to report warnings during scanner initialization. They are used in the function GetInitializeWarnings. A value of 0 indicates no warnings and a positive value indicates one of the two warnings.

If the client receives a warning and this is the first time the scanner has been used, it should be returned for repair. If the scanner has been used previously and one of these warnings appears, scanning can continue, although the results should be monitored and the scanner should be returned for repair when it is convenient.

- **INITIALIZEW_NONE**
  Indicates no warnings.

- **INITIALIZEW_EEPROM_BLANK**
  Indicates that the EEPROM's memory is blank.

- **INITIALIZEW_EEPROM_CHECKSUM_BAD**
  Indicates that the EEPROM's communications are bad.

4.16 SCAN_WARNINGS_000

These values are used to report any warnings from a scan operation. They are used in the functions GetPictureCountScanGroup and GetStripInfo. They are bitwise summed together to report all warnings. A value of 0 indicates no warnings and a positive value indicates one or more warnings. (The client will get at most one motor speed warning, but could get multiple framing warnings.)

- **SCANW_DX_GOOD**
  Indicates that enough DX codes were read successfully to determine frame numbering.

- **SCANW_DX_BAD**
  Indicates that not enough DX codes were read successfully to determine frame numbering.

- **SCANW_FILM_PRODUCT_AND_SPECIFIER_GOOD**
  Indicates that the film product and specifier codes were read successfully.

- **SCANW_FILM_PRODUCT_AND_SPECIFIER_BAD**
  Indicates that the film product and specifier codes were not read successfully.

- **SCANW_MOTOR_SPEED_GOOD**
  Indicates that the film track motor speed did not vary.

- **SCANW_MOTOR_SPEED_HALF_PERCENT_SLOW**
  Indicates that the motor speed varied by ½ percent too slow.

- **SCANW_MOTOR_SPEED_ONE_PERCENT_SLOW**
  Indicates that the motor speed varied by 1 percent too slow.

- **SCANW_MOTOR_SPEED_HALF_PERCENT_FAST**
  Indicates that the motor speed varied by ½ percent too fast.

- **SCANW_MOTOR_SPEED_ONE_PERCENT_FAST**
  Indicates that the motor speed varied by 1 percent too fast.

- **SCANW_MOTOR_SPEED_FAIR**
  Indicates that the motor speed varied by ½ percent too slow or too fast.

- **SCANW_MOTOR_SPEED_BAD**
  Indicates that the motor speed varied by 1 percent too slow or too fast.

- **SCANW_FRAMING_GOOD**
  Indicates that every picture was framed properly on the first pass.

- **SCANW_FRAMING_IN_MIDDLE**
  Indicates that one or more frames were added between other frames.

- **SCANW_FRAMING_AT_END**
  Indicates that one or more frames were added to the end of a strip.

- **SCANW_FRAMING_AT_BEGINNING**
  Indicates that one or more frames were added to the beginning of a strip.

- **SCANW_FRAMING_FAIR**
  Indicates that one or more frames were added to the beginning, middle or end of a strip.

- **SCANW_FRAMING_BAD**
  Indicates that all frames were blindly added to a strip.

- **SCANW_MAX_FILM_LENGTH_EXCEEDED**
  Indicates that the length of film scanned exceeded the maximum setting.

- **SCANW_MOF_FAILED_MAGNETICS**
  Indicates that the F235C failed to read some of the MOF data (more than 2 of 15 or 4 of 25 or 6 of 40 frames were not read).

- **SCANW_MOF_FAILED_PERFORATIONS**
  Indicates that the F235C failed to read one or more perforations.

- **SCANW_MOF_SOURCE_FILE**
  Indicates that the MOF data came from a file.

- **SCANW_MOF_SOURCE_F235C**
  Indicates that the MOF data came from the film in the F235C.

- **SCANW_MOF_FAILED**
  Indicates a MOF failure (SCANW_MOF_FAILED_MAGNETICS or SCANW_MOF_FAILED_PERFORATIONS).

- **SCANW_MOF_GROUP**
  Indicates any kind of MOF message (SCANW_MOF_FAILED or SCANW_MOF_SOURCE_FILE or SCANW_MOF_SOURCE_F235C).

4.17 MAGNETIC_DATA_STATUS_000

These values are used to report the status of the MOF data. They are used in the function GetPictureInfo2.

- **MAGNETIC_DATA_STATUS_NONE**
  The roll of film has no magnetic data, only optical.

- **MAGNETIC_DATA_STATUS_FAILED**
  The roll of film has magnetic data, but this frame's data was not read adequately.

- **MAGNETIC_DATA_STATUS_PASSED**
  The roll of film has magnetic data and this frame's data was read properly.

4.18 CALIBRATE_CONTROL_000

These values are used to control how a ForceCorrections is performed. They are used in the *iCalibrateControl* parameter. CALIBRATE_GainAndOffset and CALIBRATE_FixedPattern can be performed at the same time. CALIBRATE_Focus and CALIBRATE_FocusAdvanceFilm must be performed alone.

- **CALIBRATE_GainAndOffset**
  Indicates that a gain and offset correction should be performed.

- **CALIBRATE_FixedPattern**
  Indicates that a fixed pattern correction should be performed.

- **CALIBRATE_Focus**
  Indicates that a focus correction should be performed and film is already loaded.

- **CALIBRATE_FocusAdvanceFilm**
  Indicates that a focus correction should be performed and film is ready to be loaded.

- **CALIBRATE_LampWarmUp**
  Indicates that the scanner should warm up the lamp prior to calibrating.

- **CALIBRATE_ExerciseSteppers**
  Indicates that the scanner should exercise its stepper motors prior to calibrating.

4.19 FACTORY_RESET_CONTROL_000

These values are used to control how a ResetFactoryDefaults is performed. They are used in the *iFactoryResetControl* parameter. Both sets of parameters can be reset together by summing the two values.

- **FACTORY_RESET_Focus**
  Indicates that the focus parameters should be reset to factory defaults.

- **FACTORY_RESET_MotorSpeed**
  Indicates that the motor speed parameters should be reset to factory defaults.

4.20 FILM_TRACK_TEST_ERRORS_000

This enumeration is used to report film track test errors by way of the FilmTrackTestResults function. These errors are a result of a call to the FilmTrackTest function. This function also produces a log file called "PakonFilmTrackTestLog.txt" in the "C:\Program Files\Pakon\TLA COM Server" folder. This log file will provide more information.

- **FILM_TRACK_TEST_ERRORS_ClockTop**
  Indicates that the scanner needs to be recalibrated by a trained technician.

- **FILM_TRACK_TEST_ERRORS_DataTop**
  Indicates that the scanner needs to be recalibrated by a trained technician.

- **FILM_TRACK_TEST_ERRORS_ClockBottom**
  Indicates that the scanner needs to be recalibrated by a trained technician.

- **FILM_TRACK_TEST_ERRORS_DataBottom**
  Indicates that the scanner needs to be recalibrated by a trained technician.

- **FILM_TRACK_TEST_ERRORS_FilmEntry**
  Indicates that the film entry sensors need to be cleaned.

- **FILM_TRACK_TEST_ERRORS_FilmExit**
  Indicates that the film exit sensors need to be cleaned.

4.21 S_OR_H_000

These values are used to indicate whether a picture is marked as selected or hidden or neither. They are used in the functions GetPictureInfo, GetPictureSelection, PutPictureInfo and PutPictureSelection.

- **S_OR_H_NONE**
  Indicates that the picture is neither selected nor hidden.

- **S_OR_H_SELECTED**
  Indicates that the picture is marked as selected.

- **S_OR_H_HIDDEN**
  Indicates that the picture is marked as hidden.

4.22 SAVE_CONTROL_000

These values indicate the desired options during a save operation. They are used in the functions SaveToClientMemory and SaveToDisk.

One of the following three values must be selected:

- **SAV_SizeOriginal**
  Indicates that the picture will not be scaled, nor the bounding box rotated. This is the default (0).

- **SAV_SizeLimitForDisplay**
  Indicates that the bounding box will not be rotated to fit the picture's current rotation, it will retain its specified aspect ratio. The picture will be scaled to best fit the bounding box (unless it needs to be scaled up and SAV_DoNotScaleUp is specified).

- **SAV_SizeLimitForSave**
  Indicates that the bounding box may be rotated to best fit the picture's current rotation. The picture will also be scaled to best fit the bounding box (unless it needs to be scaled up and SAV_DoNotScaleUp is specified).

- **SAV_SizeBitMask**
  Indicates either SAV_SizeLimitForDisplay or SAV_SizeLimitForSave. This will not be used in the function call, but may be used later to determine the choice made.

Optionally, include any of the following values:

- **SAV_UseCurrentRotation**
  Indicates that the picture's current rotation will be used and the *iRotation* parameter will be ignored.

- **SAV_UseLoResBuffer**
  Indicates that the picture data should come from the low-resolution buffer and not the high-resolution buffer.

- **SAV_DoNotScaleUp**
  Indicates that the picture should not be scaled up, regardless of the bounding box.

- **SAV_UseColorKcdfs**
  Indicates that the color correction and color scene balance algorithms should use the new Kodak KCDFS library. This applies only to color negative images. Including this option with other types of images will have no effect. If this option is not included, then color negative images put through the color correction or color scene balance algorithms will use the original Pakon IMA library.

- **SAV_UseColorCorrection**
  Indicates that color correction should be done. If color correction is done, the result is a 12 bit Reference Print Density (RPD) image. Otherwise, the result is a linear transmittance (raw) image.

- **SAV_UseColorSceneBalance**
  Indicates that color scene balancing should be done. If color scene balancing is done, the result is an 8 bit standard RGB (sRGB) image. Color scene balancing cannot be done without color correction.

- **SAV_UseColorAdjustments**
  Indicates that color adjustments should be done. This will use the color settings established in PutPictureColorSettings or PutPictureColorSettingsDifferential, or the default settings will be used if neither of these were called. Color adjustments cannot be done without color scene balancing.

- **SAV_UseScratchRemovalIfAvailable**
  Indicates that the results of the Digital ICE™ scratch removal should be saved if available. It is highly recommended to include the IR channel in any save.

- **SAV_FastUpdate8BitDib**
  Color corrected images will be cached in memory. Only color scene balance and color adjustments will be done with subsequent saves. This can be used only for the SaveToClientMemory function. Also, the *iFileFormatSaveToMemory* parameter must be iFILE_FORMAT_SAVE_TO_MEMORY_DIB_8. A typical use is when the client is saving thumbnail images on screen and doing color adjustments.

- **SAV_TopDownDib**
  Indicates that the DIB should be saved right-side-up as opposed to upside-down. This can be used only for the SaveToClientMemory function.

- **SAV_FileHeader**
  Indicates that a file header should be included. This can be used only for the SaveToClientMemory function.

4.23 FILE_FORMAT_000

These values indicate what file format to use while saving pictures to disk. They are used in the function SaveToDisk.

- **iFILE_FORMAT_JPG**
  Indicates the file should be saved in JPEG format (the file extension will be .jpg).

- **iFILE_FORMAT_BMP**
  Indicates the file should be saved as a bitmap (the file extension will be .bmp).

- **iFILE_FORMAT_TIF**
  Indicates the file should be saved as a TIFF (the file extension will be .tif).

- **iFILE_FORMAT_EXIF**
  Indicates the file should be saved as an EXIF (the file extension will be .jpg).

- **iFILE_FORMAT_JPG_2000**
  Not yet implemented.

- **iFILE_FORMAT_RAW**
  Not yet implemented.

- **iFILE_FORMAT_ROLL**
  Not yet implemented.

- **iMAX_FILE_FORMATS_000**
  Indicates the number of formats defined.

4.24 FILE_FORMAT_SAVE_TO_MEMORY_000

These values indicate what file format to use while saving pictures to client memory. They are used in the function SaveToClientMemory.

- **iFILE_FORMAT_SAVE_TO_MEMORY_PLANAR_16**
  Indicates a 16 bit planar format. Use this parameter only if no color correction is being done, or if only color correction is being done. Do not use this parameter if color scene balance and/or color adjustments are being done.

- **iFILE_FORMAT_SAVE_TO_MEMORY_PLANAR_8**
  This parameter should not be used.

- **iFILE_FORMAT_SAVE_TO_MEMORY_DIB_8**
  Indicates an 8-bit DIB (device independent bitmap) format. Use this parameter if color scene balance and/or color adjustments are being done. Do not use this parameter if no color corrections are being done, or if only color corrections are being done.

4.25 ROTATE_000

These values are used to indicate how much a picture should be rotated. They are used in the functions GetPictureInfo, GetPictureRotation, PutPictureInfo, PutPictureRotation, SaveToClientMemory and SaveToDisk.

- **ROTATE_0**
  Indicates zero degrees rotation (the initial value).

- **ROTATE_90L**
  Indicates 90 degrees rotation to the left (ccw).

- **ROTATE_180**
  Indicates 180 degrees rotation.

- **ROTATE_90R**
  Indicates 90 degrees rotation to the right (cw).

- **ROTATED_90_L_OR_R**
  Indicates 90 degrees rotation left or right (ccw or cw).

- **MIRROR_L_R**
  Indicates a left | right mirrored image (this can be combined with any one of the rotation values).

4.26 SCALING_METHOD_000

These values are used to indicate the method used to scale a picture. They are used in the functions SaveToClientMemory and SaveToDisk.

- **SCALING_METHOD_NEAREST**
  This is the quickest method, but results in the lowest quality image.

- **SCALING_METHOD_BILINEAR**
  This method is medium in speed and results in a medium quality image.

- **SCALING_METHOD_BICUBIC**
  This is the slowest method, but results in the highest quality image.

4.27 COLOR_PORTRAIT_MODE_000

These values indicate whether color portrait mode is used or not. They are used in the functions GetScannerInfo000 and PutScannerInfo000. This feature is NOT YET IMPLEMENTED.

- **COLOR_PORTRAIT_MODE_NOT**
  Indicates color portrait mode will not be used.

- **COLOR_PORTRAIT_MODE_PORTRAIT**
  Indicates color portrait mode will be used.

4.28 PICTURE_LIGHTING_DIRECTION_000

These values indicate the direction of light in a particular picture. They will be used in the functions GetPictureLightingDirection and PutPictureLightingDirection. This feature is NOT YET IMPLEMENTED.

- **PICTURE_LIGHTING_DIRECTION_BACK_LIT**

- **PICTURE_LIGHTING_DIRECTION_NORMAL**

- **PICTURE_LIGHTING_DIRECTION_FRONT_LIT**

4.29 RED_EYE_000

These values indicate whether red-eye reduction is used for a particular picture or not. They will be used in the functions GetPictureRedEyeSettings and PutPictureRedEyeSettings. This feature is NOT YET IMPLEMENTED.

- **RED_EYE_AUTO**

- **RED_EYE_OFF**

- **RED_EYE_RECTANGLE**

4.30 INDEX_000

These values are used to indicate which picture to work on (besides the actual index: 0 to n-1). They are used in the functions DeletePicture, GetPictureColorSettings, GetPictureFramingInfo, GetPictureFramingUserInfo, GetPictureFramingUserInfoLowRes, GetPictureInfo, GetPictureInfo1, GetPictureInfo2, GetPictureRotation, GetPictureSelection, GetStripInfo, InsertPicture, PutPictureColorSettings, PutPictureColorSettingsDifferential, PutPictureFramingUserInfo, PutPictureInfo, PutPictureInfo1, PutPictureRotation, PutPictureSelection, SaveToClientMemory and SaveToDisk.

- **INDEX_First**
  Indicates that the first picture (0) should be used.

- **INDEX_InsertPictureAtEnd**
  Indicates that a picture should be inserted after the last picture (this is used only for the InsertPicture function).

- **INDEX_Current**
  Indicates that TLA's current picture should be used.

- **INDEX_AllSelected**
  Indicates that all pictures that are marked as selected should be used.

- **INDEX_All**
  Indicates that all pictures (except ones marked as hidden) should be used.

4.31 INT_IID_000

These values indicate the interface to use. They are used in the GetAndClearLastError function.

- **INT_IID_ITLAMain**
  Indicates TLA's Main interface.

- **INT_IID_IScanPictures**
  Indicates the Scan Pictures interface.

- **INT_IID_ISavePictures**
  Indicates the Save Pictures interface.

4.32 FRAMING_RISK_000

These values indicate the framing success for a particular picture. They are used in the function GetPictureFramingInfo. These values correspond with any framing warnings the client may get (see SCAN_WARNINGS_000) as explained below.

- **FRAMING_RISK_VERY_LOW**
  Indicates that this picture's frame was found during the scan. This corresponds to a scan warning of SCANW_FRAMING_GOOD.

- **FRAMING_RISK_LOW**
  Indicates that this picture's frame was not found initially, but there was a gap between other pictures, so a frame was added. This corresponds to a scan warning of SCANW_FRAMING_IN_MIDDLE. It also corresponds to a scan warning of SCANW_FRAMING_AT_END or SCANW_FRAMING_AT_BEGINNING if APS film was scanned and the perfs were read.

- **FRAMING_RISK_MED**
  Indicates that this picture's frame was not found initially, but there was a gap either at the beginning or end of the strip and some scan lines were present, so a frame was added. This corresponds to a scan warning of SCANW_FRAMING_AT_END or SCANW_FRAMING_AT_BEGINNING.

- **FRAMING_RISK_HIGH**
  Indicates that this picture's frame was not found initially, but there was a gap either at the beginning or end of the strip and even though no scan lines were present, aggressive framing was used, so a frame was added. This corresponds to a scan warning of SCANW_FRAMING_AT_END or SCANW_FRAMING_AT_BEGINNING.

- **FRAMING_RISK_VERY_HIGH**
  Indicates that no frames were found during the scan, so frames were blindly added. This corresponds to a scan warning of SCANW_FRAMING_BAD.

- **FRAMING_RISK_INS**
  Indicates that the user inserted this picture (its frame was not found initially). There is no corresponding scan warning.

5 Error Codes

Listed here are all the error codes available from the TLA.idl file. They are grouped into enumerations: ERROR_CODES_000, ERROR_CODES_001, etc.

5.1 ERROR_CODES_000

These error codes represent general errors that can occur any time.

- **EC_InvalidPtrToClientCallback**
  The pointer to the client callback is not valid. Check the pointer.

- **EC_WorkerThreadExists**
  A long operation within the same interface is in progress. The client must wait until the operation is complete before calling this function.

- **EC_QueryInterface**
  See Note2 below.

- **EC_CoMarshalInterThreadInterfaceInStream**
  See Note2 below.

- **EC_UnableToCreateWorkerThread**
  TLA was unable to create a worker thread. See Note2 below.

- **EC_WorkerThreadCoInitialize**
  See Note2 below.

- **EC_WorkerThreadCoGetInterfaceAndReleaseStream**
  See Note2 below.

- **EC_WorkerThreadClientSignal**
  See Note2 below.

- **EC_WorkerThreadStartTimeout**
  See Note2 below.

- **EC_ScannerNotInitialized**
  An attempt to perform some scanner function was made before the scanner was initialized. Initialize the scanner first.

- **EC_NoPicturesOrStrips**
  An attempt to perform some function on a picture or strip was made, but there are no pictures or strips. Create them first.

- **EC_TooManyRolls**
  The maximum number of rolls has been reached. Delete one or more before scanning more rolls.

- **EC_InvalidIndex**
  The index specified is out of range or cannot be used for the operation.

- **EC_InvalidMemberVariable**
  Some internal variable has an invalid value. This indicates a problem within TLA and should be reported as a software bug.

- **EC_InvalidParameter**
  One or more function parameters are not valid values.

- **EC_NoWorkerThreadForMultipleSaveToMemory**
  If saving multiple pictures, the client should have set the `bUseWorkerThread` flag.

- **EC_NoClientMemoryBuffer**
  There is no client memory buffer established. Call ClientMemoryBufferAdd first to add one or more.

- **EC_OneFileNameForMultipleSaves**
  The client is trying to save more than one picture to a file. Either save one picture at a time, or use PutPictureInfo or PutPictureInfo1 to specify a file for each picture (and use NULL as the file name in this function).

- **EC_StartUpError**
  Indicates a problem with TLA receiving services from the operating system. Try rebooting the host computer.

- **EC_CBAdviseAlreadyCalled**
  The CBAdvise function has already been called, it needs to be called only once.

- **EC_CBAdviseNotCalled**
  The CBAdvise function has not been called, it needs to be called first.

- **EC_InitializeScannerAlreadyCalled**
  The InitializeScanner function has already been called, it needs to be called only once.

- **EC_AdjustMotorSpeedIsZero**
  Not yet implemented.

- **EC_NotSupportedByHW**
  Indicates that the scanner's DX software and/or hardware is too old to support a film track test.

- **EC_PreviousError**
  Used in the call stack trace to indicate part of an error previously listed in the trace.

- **EC_FileNameListEmpty**
  No file names were provided. At least one must be listed.

- **EC_LampError**
  Indicates a lamp error (low voltage or high temperature or slow fan or burn out).

- **EC_ChangingFrameNumberWithAps**
  Indicates that an attempt to change the frame number of APS film was made. This is not allowed.

- **EC_NotAllowedWithAps**
  Indicates that an attempt to insert a picture was made with APS film. This is not allowed.

5.2 ERROR_CODES_001

These error codes also represent general errors that can occur any time. For error codes that begin "EC_WIN_", these indicate TLA having trouble getting services from the Windows operating system.

- **EC_AidNoRoll**
  Indicates a programming error within TLA.

- **EC_ApsCartridgeUnpacked**
  Indicates that an APS cartridge with extracted film was found in the scanner when it was powered up. The film may be jammed.

- **EC_ApsEjectButtonPressed**
  Indicates that the APS eject button was pressed when film was out of the cartridge (e.g. during a scan or during retraction).

- **EC_ApsFilmEndError**
  Indicates that during an APS scan, the film's last perforation was not detected. The scanner's perf sensor may be bad.

- **EC_ApsFilmJamExtract**
  Indicates that during an APS scan, the film stopped (before it reached the light bar) and is probably jammed.

- **EC_ApsFilmJamScan**
  Indicates that during an APS scan, the film stopped and is probably jammed.

- **EC_ApsFilmJamRetract**
  Indicates that during an APS scan or a manual retract, the film stopped (while retracting) and is probably jammed.

- **EC_ApsNoCartridge**
  Indicates that an APS cartridge was not present when extraction was requested.

- **EC_ApsOverflow**
  Indicates that a data overflow condition occurred while reading MOF data.

- **EC_ApsPark**
  Indicates that, after retraction, the APS cartridge could not be parked (the film status indicator could not be returned to its correct position).

- **EC_ApsParkInit**
  Indicates that the APS park sensor could not be initialized.

- **EC_ApsUnprocessedFilm**
  Indicates that the IPI tab on the cartridge is not broken, signifying that the film was not processed.

- **EC_BadFileData**
  Color correction information could not be read from file. Try reinstalling TLA.

- **EC_BadSimulatorFile**
  Indicates that the scan file used for the simulator version of TLA is missing or corrupted.

- **EC_BufferDriveMegabytesRollTooSmall**
  Indicates that the size of the high-resolution buffer for MB per roll is too small. Increase the buffer size.

- **EC_ClientMemoryBufferInUse**
  Indicates that a client memory buffer is in use. Wait until the SaveToClientMemory function finishes, then try again.

- **EC_CreateDirectoryInvalidForm**
  Indicates that the target directory is not in valid form. Call PutSaveInfo to fix it.

- **EC_CS_DoubleUnlock**
  Not yet implemented.

- **EC_CS_InvalidUnlock**
  Not yet implemented.

- **EC_CS_NotUnlockedAtExit**
  Not yet implemented.

- **EC_DllInitialize**
  Indicates that the scratch removal DLL could not be initialized. Try reinstalling TLA.

- **EC_EEPromAddress**
  Indicates an internal problem with the scanner's EEPROM. Call a technician for repair.

- **EC_EEPromCorrupted**
  Indicates an internal problem with the scanner's EEPROM. Call a technician for repair.

- **EC_EEPromLength**
  Indicates an internal problem with the scanner's EEPROM. Call a technician for repair.

- **EC_EEPromMemoryAddress**
  Indicates an internal problem with the scanner's EEPROM. Call a technician for repair.

- **EC_EEPromWarningBlank**
  Indicates an internal problem with the scanner's EEPROM. Call a technician for repair.

- **EC_EEPromWarningCheckSumBad**
  Indicates an internal problem with the scanner's EEPROM. Call a technician for repair.

- **EC_FileNotFound**
  Indicates that some TLA file could not be found during startup or during "Import from File". Try reinstalling TLA.

- **EC_FilmInGuides**
  Indicates that during the scanner's power-on self-test, film was detected in the film guide.

- **EC_FirmwareVerification**
  Indicates that a firmware download failed. Try again or call a technician for repair.

- **EC_FocusCurvatureThreshold**
  Indicates that focus is out of alignment. Correct it with a call to ForceCorrections. If this fails, use Calibration Wizard.

- **EC_FocusOutsideRegionOfInterest**
  Indicates that focus is out of alignment. Correct it with a call to ForceCorrections. If this fails, use Calibration Wizard.

- **EC_FocusPredictorThreshold**
  Indicates that focus is out of alignment. Correct it with a call to ForceCorrections. If this fails, use Calibration Wizard.

- **EC_FocusQuadRegress**
  Indicates that focus is out of alignment. Correct it with a call to ForceCorrections. If this fails, use Calibration Wizard.

- **EC_HardwareFault**
  Indicates a polling error with a scanner controller. Check the logs for more information. If severe, call a technician for repair.

- **EC_ImageNotPlanar**
  Indicates that the file format specified is not planar and needs to be.

- **EC_ImportedFileColor**
  Indicates that the number of channels specified in the imported file is invalid. It must be one (b&w) or three (color).

- **EC_InsufficientMemoryForSaveToMemory**
  Indicates that the client memory buffer is too small to hold the picture. Recalculate and create a new buffer.

- **EC_InsufficientMemoryPassedIn**
  Indicates a programming error within TLA and should be reported as a software bug.

- **EC_LampWarmUpFailure**
  Indicates that the lamp failed to reach a stable state during warm up.

- **EC_MemoryNew**
  Indicates a problem with TLA receiving services from the operating system. Try rebooting the host computer.

- **EC_MissingDllFunction**
  Indicates that a scratch removal DLL function could not be found. Try reinstalling TLA.

- **EC_NoFixedPatternCorrection**
  Indicates a programming error within TLA and should be reported as a software bug.

- **EC_NoHighResolutionBuffer**
  Indicates a programming error within TLA and should be reported as a software bug.

- **EC_NoStripsScanned**
  Indicates that the scanner timed out before film was detected. The user should restart the scan and insert the film before the *NoFilmTimeout* (see GetScannerInfo000).

- **EC_ParsingError**
  During a firmware download, the hex file could not be parsed. Try replacing the file.

- **EC_PicVersion**
  Indicates that the version of some firmware within the scanner is out of date. Update the firmware to the latest versions.

- **EC_PreviousHardwareFaultAps**
  Indicates that there was some APS error reported earlier and the error logs should be checked.

- **EC_ProcessedRingTailOverflow**
  Indicates that there was a breakdown in the communication of scan lines from the scanner to TLA. The user can retry the scan.

- **EC_RegistryRead**
  Indicates that TLA could not read a registry value. Try reinstalling TLA.

- **EC_ScanLineAcquisition**
  Indicates that TLA has trouble communicating with the scanner. Check the USB cable and connections, otherwise the scanner will need to be diagnosed and repaired by a technician.

- **EC_SelfTestFailedCcdStepper**
  Indicates that the self-test on the CCD stepper motors failed. Call a technician for repair.

- **EC_SelfTestFailedFilmDrive**
  Indicates that the self-test on the film drive failed. Call a technician for repair.

- **EC_SelfTestFailedLensStepper**
  Indicates that the self-test on the lens stepper motors failed. Call a technician for repair.

- **EC_StepperAlreadyMoving**
  Indicates a programming error within TLA and should be reported as a software bug.

- **EC_StepperDidNotStop**
  Indicates a malfunctioning stepper motor. A service technician should be called to troubleshoot the error.

- **EC_StepperPosition**
  Not yet implemented.

- **EC_SystemInfo**
  Indicates that the computer processor is not an Intel/AMD Pentium or better and does not meet Pakon's specs. Check the specifications and replace the CPU or computer.

- **EC_TimeOut**
  Indicates some sort of timeout. If saving multiple pictures to client memory, a timeout may occur if a buffer isn't ready (use InitializeScanner to set this timeout). A timeout can also occur if there's a problem with the TLA configuration and may be remedied by cycling power on the scanner, or rebooting the computer, or reinstalling TLA. Finally, a timeout can occur if TLA has trouble receiving services from the operating system. Try rebooting the host computer.

- **EC_WrongByteCount**
  Indicates a problem with TLA receiving services from the operating system. Try rebooting the computer to clear the error.

- **EC_WIN_CancelIo**
  See Note2 below.

- **EC_WIN_CreateDirectory**
  See Note2 below.

- **EC_WIN_CreateEvent**
  See Note2 below.

- **EC_WIN_CreateFileMapping**
  See Note2 below.

- **EC_WIN_DeviceIoControl**
  See Note2 below.

- **EC_WIN_FileClose**
  See Note2 below.

- **EC_WIN_FileOpen**
  See Note2 below.

- **EC_WIN_FileRead**
  See Note2 below.

- **EC_WIN_FileSetPointer**
  See Note2 below.

- **EC_WIN_FileTimeToSystemTime**
  See Note2 below.

- **EC_WIN_FileWrite**
  See Note2 below.

- **EC_WIN_FindFirstFile**
  See Note2 below.

- **EC_WIN_FreeLibrary**
  See Note2 below.

- **EC_WIN_GetDiskFreeSpace**
  See Note2 below.

- **EC_WIN_GetDiskFreeSpaceEx**
  See Note2 below.

- **EC_WIN_GetFileSize**
  During a firmware download, the hex file was the wrong size. Try replacing the file.

- **EC_WIN_GetOverlappedResult**
  See Note2 below.

- **EC_WIN_LoadLibrary**
  See Note2 below.

- **EC_WIN_MapViewOfFile**
  Not yet implemented.

- **EC_WIN_OpenEvent**
  Not yet implemented.

- **EC_WIN_OpenFileMapping**
  Not yet implemented.

- **EC_WIN_ResetEvent**
  See Note2 below.

- **EC_WIN_SetEndOfFile**
  See Note2 below.

- **EC_WIN_SetEvent**
  See Note2 below.

- **EC_WIN_SetFilePointerEx**
  See Note2 below.

- **EC_WIN_SetProcessWorkingSetSize**
  See Note2 below.

- **EC_WIN_UnmapViewOfFile**
  Not yet implemented.

- **EC_WIN_VirtualAlloc**
  See Note2 below.

- **EC_WIN_VirtualFree**
  See Note2 below.

- **EC_WIN_VirtualLock**
  See Note2 below.

- **EC_WIN_VirtualUnlock**
  See Note2 below.

- **EC_WIN_WaitForSingleObject**
  See Note2 below.

- **EC_DXPotsWillNotAdjust**
  Used for calibration only.

- **EC_DXNoFilmFound**
  During a film track test, no film was inserted in the time allowed. Run the test again and insert film more quickly.

- **EC_DXNoGoodBrightSpotFound**
  Used for calibration only.

- **EC_DXAdjustingPotsForGoodSignal**
  Used for calibration only.

- **EC_DXBadSwing**
  During a film track test, the scanner could not get a good voltage swing on a DX sensor.

- **EC_NoFilmTimeOut**
  During a scan, the scanner timed out waiting for film. Either the film is jammed or the operator did not insert the film before the timeout expired.

5.3 ERROR_CODES_010

These error codes all have to do with the F235 driver. This includes the chain of communications between TLA and the scanner, including the USB board in the scanner, the firmware in this USB board, the USB cable, the USB card in the computer, the USB driver for this card, and the F235 driver on the host computer.

- **EC_DRV_Unknown**
  Not yet implemented.

- **EC_DRV_CannotFindStartOfScanLine**
  See Note3 below.

- **EC_DRV_RingTailOverflow**
  See Note3 below.

- **EC_DRV_LostSync**
  See Note3 below.

- **EC_DRV_InvalidPacketType**
  See Note3 below.

- **EC_DRV_PacketBusy**
  See Note3 below.

- **EC_DRV_FifoOverflow**
  See Note3 below.

- **EC_DRV_PacketChecksumErr**
  See Note3 below.

- **EC_DRV_PacketOverFlowErr**
  Not yet implemented.

- **EC_DRV_PacketCommErr**
  See Note3 below.

- **EC_DRV_PacketCmdErr**
  See Note3 below.

- **EC_DRV_PacketHostErrorNoAck**
  See Note3 below.

- **EC_DRV_PacketHostErrorFormat**
  See Note3 below.

- **EC_DRV_PacketHostErrorCkSum**
  See Note3 below.

- **EC_DRV_PacketHostErrorEndPointFormat**
  See Note3 below.

- **EC_DRV_PacketHostErrorEndPointTimeOut**
  See Note3 below.

- **EC_DRV_PacketHostErrorEndPointLength**
  See Note3 below.

- **EC_DRV_PacketHostErrorAlgo**
  See Note3 below.

- **EC_DRV_PacketHostErrorBus**
  See Note3 below.

- **EC_DRV_PacketHostErrorUndefined**
  See Note3 below.

- **EC_DRV_PacketReadWriteMismatch**
  See Note3 below.

- **EC_DRV_InfoBufferString**
  See Note3 below.

- **EC_DRV_TransferInProgress**
  See Note3 below.

5.4 ERROR_CODES_020

These error codes all have to do with the color correction and scene balance algorithms within TLA.

- **EC_PI_UNKNOWN**
  See Note1 below.

- **EC_PI_MEMORY**
  See Note2 below.

- **EC_PI_CANT_WRITE_PAKONERRORLOGPI**
  See Note2 below.

- **EC_PI_CR_INPUT_PROFILE**
  Indicates a problem locating the necessary files on the host computer that support the color correction data path. The problem may be remedied by reinstalling TLA or manually adjusting registry values.

- **EC_PI_INPUT_PROFILE**
  Indicates a problem locating the necessary files on the host computer that support the color correction data path. The problem may be remedied by reinstalling TLA or manually adjusting registry values.

- **EC_PI_OUTPUT_PROFILE**
  Indicates a problem locating the necessary files on the host computer that support the color correction data path. The problem may be remedied by reinstalling TLA or manually adjusting registry values.

- **EC_PI_RPD2ROMM_PROFILE**
  Indicates a problem locating the necessary files on the host computer that support the color correction data path. The problem may be remedied by reinstalling TLA or manually adjusting registry values.

- **EC_PI_NO_MMX_PROCESSOR**
  Indicates a problem with TLA receiving services from the operating system. Try rebooting the computer to clear the error.

- **EC_PI_MIN_MAX_RANGE_RED**
  See Note1 below.

- **EC_PI_MIN_MAX_RANGE_GREEN**
  See Note1 below.

- **EC_PI_MIN_MAX_RANGE_BLUE**
  See Note1 below.

- **EC_PI_MIN_MAX_RANGE_BRIGHTNESS**
  See Note1 below.

- **EC_PI_MIN_MAX_RANGE_CONTRAST**
  See Note1 below.

- **EC_PI_INVALID_ORIENTATION**
  See Note1 below.

- **EC_PI_ERROR_SETTING_LOCKBEAM**
  See Note1 below.

- **EC_PI_INVALID_FILE_FORMAT**
  See Note1 below.

- **EC_PI_COMBINE_INPUT_OUTPUT_PROFILE**
  See Note1 below.

- **EC_PI_CR_COMBINE_INPUT_OUTPUT_PROFILE**
  See Note1 below.

- **EC_PI_KNOWN_EXCEPTION_RECORDED**
  If this occurs while opening a file, the file may be corrupt or invalid. Also see Note1.

- **EC_PI_UNKNOWN_EXCEPTION_RECORDED**
  See Note1 below.

- **EC_PI_CR_LUTS6**
  See Note1 below.

- **EC_PI_KCDFS_INIT_FAILED**
  This error code is not yet used.

5.5 ERROR_CODES_030

These error codes all have to do with the buffer drive (N:), also known as the Pakon File System.

- **EC_PFS_PartitionAlreadySelected**
  See Note1 below.

- **EC_PFS_FileSystemExists**
  While trying to open a partition, a file system was detected. The function exits so as not to destroy user data.

- **EC_PFS_PartitionSelected**
  See Note1 below.

- **EC_PFS_FileSystemNotEmpty**
  See Note1 below.

- **EC_PFS_NullFilePointer**
  See Note1 below.

- **EC_PFS_FileAlreadyDeleted**
  See Note1 below.

- **EC_PFS_FilePointerDeleted**
  See Note1 below.

- **EC_PFS_ReadPastEOF**
  See Note2 below.

- **EC_PFS_NotLastStripInFile**
  See Note1 below.

- **EC_PFS_FileLengthNotSet**
  See Note1 below.

- **EC_PFS_FileSystemFull**
  The buffer drive (N:) is full. Delete buffers or increase the partition size.

- **EC_PFS_WriteSizeInvalid**
  See Note1 below.

- **EC_PFS_WritePastEOF**
  See Note1 below.

- **EC_PFS_BadDrive**
  The buffer drive (N:) is bad. Mark the bad sectors of the drive or replace it.

- **EC_PFS_InvalidPointer**
  See Note1 below.

- **EC_PFS_WritingToCompletedStrip**
  Not yet implemented.

- **EC_PFS_NotEnoughDiskSpace**
  The client is trying to create a swap file that is too large for the partition. Increase the partition size or decrease the HiResMegabytesRoll registry value.

5.6 ERROR_CODES_040

These error codes all have to do with the digital ICE scratch removal system.

- **EC_DICE_MemErr**
  See Note2 below.

- **EC_DICE_InProgress**
  See Note1 below.

- **EC_DICE_NotInProgress**
  See Note1 below.

- **EC_DICE_InvalidThread**
  See Note1 below.

- **EC_DICE_InvalidParameter**
  See Note1 below.

- **EC_DICE_QueueFull**
  See Note1 below.

- **EC_DICE_InternalErr**
  See Note1 below.

- **EC_DICE_CodeErr**
  See Note1 below.

- **EC_DICE_CouldNotLoad**
  Indicates that TLA could not load the scratch removal library. Try reinstalling TLA.

- **EC_DICE_Unknown**
  See Note1 below.

5.7 Error Notes

Note1: These errors indicate an internal TLA software error. Reboot and/or reinstall TLA. If problems persist, report the bug to Pakon.

Note2: These errors indicate a problem with TLA receiving services from the operating system. Try rebooting the host computer to clear the error.

Note3: These errors indicate a breakdown in the communications link between TLA and the scanner via the USB bus. Verify that the USB cable is the proper length and defect free. Try cycling power on the scanner and restarting TLA. Try reinstalling the USB driver. Lastly, try replacing the USB boards.