Imaging Toolkit

Integration Guide for DLL

January 2021

www.lexmark.com

Contents

Overview

Imaging Toolkit integrates scanning capabilities directly to an application without using Microsoft Windows Imaging Architecture or TWAIN data source frameworks.

When Imaging Toolkit is called by a host program, a scan profile is sent to the designated multifunction product (MFP). A listening port on the host computer remains open until the user scans using the profile. When scanning, data is sent to Imaging Toolkit, and then passes the scan job back to the calling program.

Imaging Toolkit consists of a dynamic link library (DLL) that provides the imaging functionality, header, and library files necessary to use the DLL with your application. It also includes documentation of the DLL API. You can use the DLL with C++ applications, Visual Basic applications, and any other Windows application that can call a DLL.

Integrating Imaging Toolkit involves the following components:

- Creating an application
- Executing the application

Files included with Imaging Toolkit and their descriptions

File or folder	Description
Prerequisites	This folder contains the following: • **.Net Framework 4.5 update 2**—Need to run this file to install .net framework which is necessary to use the Toolkit DLL. • **C++ redistributable for Visual Studio 2012**—Need to run this file to install C ++ redistributable component.
SampleApplication	Contains one Sample application by using API calls from the Imaging Toolkit DLL.
integration_guide_dll.pdf	Integration guide for Windows development environment.
License.txt	This file contains the *End User License Agreement*.
LIT_native_sdk_32bit	This folder contains the following: • **bin** folder—The basic set of binary files to run Imaging Toolkit from the command line, with properties files that can be used to change scan properties. It also contains a sample GUI application. • **include** folder—The “include” file required to use the DLL with a C++ application. • **lib** folder—The library file required to use the DLL with a C++ application.
LIT_native_sdk_64bit	This folder contains the following: • **bin** folder—The basic set of binary files to run Imaging Toolkit from the command line, with properties files that can be used to change scan properties. It also contains a sample GUI application. • **include** folder—The “include” file required to use the DLL with a C++ application. • **lib** folder—The library file required to use the DLL with a C++ application.

File or folder	Description
ReadMe.txt	This file contains the following: • Technical updates • Supported printers • Known issues

Supported operating systems

- Microsoft Windows 10 Enterprise
- Microsoft Windows 8 Enterprise
- Microsoft Windows 7 Ultimate
- Microsoft Windows Server 2016 Standard
- Microsoft Windows Server 2012 R2 Standard
- Microsoft Windows Server 2008 R2 EE

Configuring the application

Contents of the bin directory

The following bin folders contain binary files, including the Imaging Toolkit DLL, supporting DLLs, and a sample application:

- **LIT_native_SDK_64bit\bin** for 64-bit operating systems
- **LIT_native_SDK_32bit\bin** for 32-bit operating systems

64-bit DLL bin and 32-bit DLL bin directory files and their descriptions

File	Description
FirebirdSql.Data.FirebirdClient.dll	A required file that must be copied to the same directory as mfpsb.dll. This file is a third-party DLL that is used for communicating with the Firebird® database.
LITScan.dll	A required file that must be copied to the same directory as mfpsb.dll.
LITScanManaged.dll	A required file that must be copied to the same directory as mfpsb.dll.
mfpsb.dll	A file that provides imaging functionality.
MFPSB_Sample.exe	A sample UI that is used to test Toolkit functionality.
Newtonsoft.Json.dll	A required file that is used for JavaScript Object Notation data parsing.
ObjectStoreLibrary.dll	A required file that must be copied to the same directory as mfpsb.dll. This file is used to communicate with the printer.
Conf/lddtomfp2.xsl	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This style sheet is used to add profiles on a printer.
Conf/lddtomfp3.xsl	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This style sheet is used to add profiles on a printer.
Conf/lddtomfp4.xsl	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This style sheet is used to add profiles on a printer.
Conf/lddtomfp5.xsl	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This style sheet is used to add profiles on a printer.
Conf/lddtomfp6.xsl	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This style sheet is used to add profiles on a printer.
Conf/LexmarkImagingToolkitForMoja.properties	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This properties file is used to change the scan settings of a profile. This file is applicable to printers running on eSF version 5 or later.

File	Description
Conf/LexmarkImagingToolkitForWC.properties	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This properties file is used to change the scan settings of a profile. This file is applicable to to printers running on eSF version 4 or earlier.
Config/AppSettings.xml	A required file that must be copied to the same directory as mfpsb.dll in the Conf folder. This file is used for printer communication settings.
Images/image_dn_MB.gif	A required file that must be copied to the same directory as mfpsb.dll in the Images folder. This file is the default profile icon shown on the printer home screen.
Images/image_dn_MC.gif	A required file that must be copied to the same directory as mfpsb.dll in the Images folder. This file is the default profile icon shown on the printer home screen.
Images/image_up_MB.gif	A required file that must be copied to the same directory as mfpsb.dll in the Images folder. This file is the default profile icon shown on the printer home screen.
Images/image_up_MC.gif	A required file that must be copied to the same directory as mfpsb.dll in the Images folder. This file is the default profile icon shown on the printer home screen.

Understanding the properties file for setting defaults

The LexmarkImagingToolkit.properties file stores the configuration settings for Imaging Toolkit. This file must be located in the Conf subdirectory where mfpsb.dll is stored. If Imaging Toolkit cannot find the LexmarkImagingToolkit.properties file, then it automatically uses the default settings. Application properties are set using the following keys:

Note: Key values are case-sensitive.

Key	Default value	Description
BRIGHTNESS[1] or **darkness**[2]	**4**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **BRIGHTNESS**. Adjust the darkness on the scanned image by specifying **BRIGHTNESS+1**.
COMPRESSION	**JPEG**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **COMPRESSION**.
CONTRAST	**MIXED**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **CONTRAST**.

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

Key	Default value	Description
DEPTH[1] or **imageComposition**[2]	**8**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **DEPTH**.
DUPLEX[1] or **scanDuplex**[2]	**SINGLE**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **DUPLEX**.
JOBBUILD	**FALSE**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **JOBBUILD**.
LINEARXFER	**FALSE**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **LINEARXFER**.
ORIENTATION	**PORTRAIT**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **ORIENTATION**.
RESOLUTION	**150**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **RESOLUTION**.
SCANPREVIEW	**FALSE**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **SCANPREVIEW**.
SHORTCUT[1] or **shortcutId**[2]	**0**	The default setting used when a shortcut is not specified.
SIZE[1] or **mediaSize**[2]	**LETTER**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **SIZE**.
TYPE[1] or **fileFormat**[2]	**JPEG**	The default setting used for scan jobs when **MFPSB_SetScanSetting()** is not called for **TYPE**.

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

Registry setting	Default value	Description
DefaultPort	**9750**	The default port used to communicate to a printer.
FirstAvailableShortcut	**TRUE**	If this setting is set to **TRUE**, then Imaging Toolkit creates a profile with the next available shortcut on the printer.
MaxShortcut	**16535**	An integer representing the largest number assigned as a profile shortcut.
MinShortcut	**1**	An integer representing the smallest number assigned as a profile shortcut.
NumConsecutiveRetryPorts	**10**	If **SinglePort** is set to **FALSE**, then this setting is the number of attempts to resend data when the default port is busy.
SinglePort	**FALSE**	If this setting is set to **TRUE**, then all uploaded scan profiles use the port number specified in **DefaultPort**. If this setting is set to **FALSE**, then each profile uses a different port.

Understanding the scan settings

Notes:

- Depending on your printer model, some settings may vary.
- The scan settings are case-sensitive.
- To avoid errors, make sure that the values for the **RESOLUTION**, **COMPRESSION**, **DEPTH**, and **TYPE** settings are compatible. For more information on incompatible scan settings, see "Incompatible scan settings" on page 13.
- For more information on scan settings and their supported values for printers running on eSF version 5 or later, navigate to the **/webservices/netscan/v1/info/scanner** page of your printer.

Setting	Accepted values
BACKGROUNDREMOVAL	-4 to 4
BRIGHTNESS[1]	0 to 8
darkness[2]	1 to 9
COLORBALANCEBLUE **Note:** Depending on your printer model, this setting may not be available.	-5 to 5
COLORBALANCEGREEN **Note:** Depending on your printer model, this setting may not be available.	-5 to 5
COLORBALANCERED **Note:** Depending on your printer model, this setting may not be available.	-5 to 5
COMPRESSION	• **G31D** • **G32D** • **G4** • **JPEG**[3] • **PACKBITS** • **ZLIB** • **LZW** • **NONE**

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

[3] Indicates the default value.

Setting	Accepted values
CONTENT[1] or **contentType**[2]	• **MIXED** • **PHOTO** • **PHOTO CONTONE** • **TEXT** • **GRAPHIC** **Note:** This setting is optional. The scan quality can also be customized by changing the contrast.
CONTRAST	• 0 to 5 • **MIXED**[3] • **PHOTO** • **PHOTO CONTONE** • **TEXT** • **GRAPHIC**
DEPTH[1]	• **1** • **8** • **24**
imageComposition[2]	• **BW1BIT** • **GRAY8BIT** • **RGB24BIT**
DOCUMENTSOURCE	• **BLACK AND WHITE LASER**[3] • **COLOR LASER** • **INKJET** • **PHOTOGRAPH** • **MAGAZINE** • **NEWSPAPER** • **PRESS** • **GENERAL OFFICE** **Note:** This setting is optional. It is required only if **CONTENT** is set to **GRAPHIC**.
DUPLEX[1] or **scanDuplex**[2]	• **SINGLE**[3] or **SIMPLEX** • **BOTH** • **DUPLEXSHORTEDGE** • **DUPLEXLONGEDGE**
EDGETOEDGE	• **TRUE** • **FALSE**[3]

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

[3] Indicates the default value.

Setting	Accepted values
INVERTIMAGE[1] or **negativeImage**[2]	• **TRUE** • **FALSE**[3]
JOBBUILD	• **TRUE** • **FALSE**[3]
JPEGQUALITY	5 to 90 (**80**[3])
LINEARXFER **Note:** Depending on your printer model, this setting may not be available.	• **TRUE** • **FALSE**[3]
MIRROR IMAGE	• **TRUE** • **FALSE**[3]
MULTIPAGETIFF **Note:** Depending on your printer model, this setting may not be available.	• **TRUE** • **FALSE**[3]
ORIENTATION	• **PORTRAIT**[3] • **LANDSCAPE**
PDFVERSION **Note:** Depending on your printer model, this setting may not be available.	• **1.2** • **1.3** • **1.4** • **1.5** • **1.6** • **A-1a**
RESOLUTION	• **75** • **150**[3] • **200** • **300** • **400** • **600** • **1200**
SCANPREVIEW	• **TRUE** • **FALSE**[3]
SCANRECOVERY **Note:** Depending on your printer model, this setting may not be available.	• **TRUE** • **FALSE**[3]

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

[3] Indicates the default value.

Setting	Accepted values
SHADOWDETAIL	-4 to 4
SHORTCUT[1] or **shortcutId**[2]	**0**[3] **For FirstAvailableShortcut=FALSE** • If the shortcut value is **0**, then Imaging Toolkit creates a profile without any shortcut. • If the shortcut value is in use, then a profile is not created. • If the shortcut value is available, then a profile is created with the specified shortcut value. **For FirstAvailableShortcut=TRUE** • If the shortcut value is **0**, then a profile is created with any available shortcut value from a specified range. • If the shortcut specified is in use, then a profile is created with any available shortcut value from a specified range. • If the shortcut specified is available, then a profile is created with the specified shortcut value.

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

[3] Indicates the default value.

Setting	Accepted values
SIZE[1] or **mediaSize**[2]	• **11x17** • **3x5 PHOTO** • **4x6 PHOTO** • **A3** • **A4** • **A5** • **A6** • **AUTO SIZE** • **B3** • **B4** • **B5** • **BOOK** • **BOOK ORIGINAL** (same as **BOOK**) • **BUSINESS CARD** • **EXECUTIVE** • **FOLIO** • **ID-1** • **LEGAL** • **LETTER**[3] • **MIXED SIZES** • **OFICIO** • **STATEMENT** • **TABLOID** (same as **11x17**) • **UNIVERSAL**
SOURCE[1] or **scanSource**[2]	• **ADF** • **FLATBED** • **ANY** • **ADF1PAGE**
TYPE[1] or **fileFormat**[2]	• **JPEG**[3] • **TIFF** • **PDF** • **PS** • **RAW**

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

[3] Indicates the default value.

Incompatible scan settings

Some scan settings do not work well in combination, and results vary. While others may exist, the following combinations are known to be incompatible:

Scan setting	Does not work with
COMPRESSION = NONE	• TYPE[1] or fileFormat[2] = JPEG • TYPE[1] or fileFormat[2] = TIFF • TYPE[1] or fileFormat[2] = PS • TYPE[1] or fileFormat[2] = RAW
COMPRESSION = JPEG	TYPE[1] or fileFormat[2] = TIFF
COMPRESSION = ZLIB	• TYPE[1] or fileFormat[2] = JPEG • TYPE[1] or fileFormat[2] = TIFF • TYPE[1] or fileFormat[2] = PS • TYPE[1] or fileFormat[2] = RAW
COMPRESSION = PACKBITS	• TYPE[1] or fileFormat[2] = JPEG • TYPE[1] or fileFormat[2] = PDF • TYPE[1] or fileFormat[2] = PS • TYPE[1] or fileFormat[2] = RAW
COMPRESSION = LZW	• TYPE[1] or fileFormat[2] = JPEG • TYPE[1] or fileFormat[2] = PDF • TYPE[1] or fileFormat[2] = PS • TYPE[1] or fileFormat[2] = RAW
DEPTH[1] or imageComposition[2] = 1	TYPE[1] or fileFormat[2] = JPEG
RESOLUTION = 1200	Not supported; results vary. Some printers default to a scan resolution of 150 dpi if this value is used.

[1] Supported by printers running on eSF version 4 or earlier.

[2] Supported by printers running on eSF version 5 or later.

Custom scan sizes

Size	Actual size of scanned image
ID CARD	Custom Scan Size 1
CUSTOM SCAN SIZE 2	Custom Scan Size 2
CUSTOM SCAN SIZE 3	Custom Scan Size 3
CUSTOM SCAN SIZE 4	Custom Scan Size 4
CUSTOM SCAN SIZE 5	Custom Scan Size 5
CUSTOM SCAN SIZE 6	Custom Scan Size 6

Note: Size names are not case-sensitive.

You can also rename Custom Scan Sizes, or specify new scan size settings. Imaging Toolkit treats the renamed Custom Scan Size as a valid paper size name while scanning. To change Custom Scan Size names or settings, do the following:

1. Access the printer Embedded Web Server.
2. Click **Settings** > **Paper Menu** > **Custom Scan Sizes**.
3. Click **ID Card** or **Custom Scan Size** (1–6).
4. From the Settings page, specify the following:
   - **Scan Size Name**—Leave blank to use the default label, or type a new name for your custom scan (maximum of 64 characters).

     Note: When using a standard name such as **`Letter`** or **`Legal`** as a Custom Scan Size name, Imaging Toolkit uses the dimensions specified in Custom Scan Settings.
   - **Width**—Specify the width of your scan, in inches. The accepted values are from 1.00 to 8.50.
   - **Height**—Specify the height of your scan, in inches. The accepted values are from 1.00 to 14.00.
   - **Orientation**
   - **2 scans per side**—Enable the printer to scan two images on the same side of the scanned output.
5. Click **Submit**.

   Note: Clicking **Reset Form** clears the settings.

Understanding the created file types

Image file

The image file returned after the scan job is completed has the same base name as the file name passed to the DLL file.

If the file type does not support multiple-page files such as JPEG, then one file is returned for each page of the scan. The image contains the same base name as the file name passed to the DLL file, followed by the page number of the scan. For example, image1.jpg.

Using the application API

Required functions

Imaging Toolkit lets applications perform the imaging process by only making function calls. The API version of Imaging Toolkit consists of 16 functions. Depending on your application, some functions may not be necessary. Imaging Toolkit supports profile names and file names that can contain ISO-8859-1 characters.

To start the imaging process, create a profile, and then receive the scanned image.

Creating a profile

A profile describes scan job parameters such as the contrast, depth, duplex, orientation, resolution, size, and type. The profile also determines where the scanned image is sent using an IP address and a TCP port number.

As part of creating a profile, a socket must be created to listen for a connection on the TCP port specified by the profile.

Note: Make sure that the socket is created *before* you send the profile to the printer. This way, the socket can be bound to the exact port number that was sent to the printer.

Receiving a scanned file

When selecting the profile on the printer control panel, the printer connects to the socket and sends the images, along with metadata. Metadata is not part of the actual image data, but is an information used as a communication tool between the printer and the computer.

For example, when scanning more than one page in a format that does not allow multiple pages, the printer sends each page as a separate file. Each file contains metadata that says that the file is complete, and another file is starting. When the printer is done sending the images, it closes the connection and the process is complete.

Creating an application

1. Initialize Imaging Toolkit. For more information, see “Understanding Imaging Toolkit initialization” on page 16.
2. Create a context. For more information, see “Understanding context creation” on page 16.
3. If applicable, set the port range. For more information, see “Understanding the port range” on page 16.
4. Set the scan settings. For more information, see “Understanding scan settings specification” on page 17.
5. Set the output directory and filename.
6. If applicable, register a callback function. For more information, see “Registering a callback function” on page 19.
7. Call one of the scan functions. For more information, see “Calling the scan functions” on page 19.

8 Uninitialize the Imaging Toolkit. For more information, see "Uninitializing" on page 19.

9 Retrieve the created scan context. For more information, see "Understanding scan context retrieval" on page 20.

Note: All examples in the following sections are shown using C++.

Understanding Imaging Toolkit initialization

```
MFPSB_Initialize()
```

The **`MFPSB_Initialize()`** function lets the Imaging Toolkit library initialize internal data structures, read the Windows registry, and configure Imaging Toolkit. If **`SinglePort`** is set to **`TRUE`**, then the function also creates a receiving socket. We recommend calling this API when initializing the application, if **`SinglePort`** is set to **`TRUE`**.

Note: If the call to initialize the library is skipped, then the first call to any of the other API functions is used to initialize the library.

Understanding context creation

```
MFPSB_CreateScanContext()
```

The **`MFPSB_CreateScanContext()`** function returns an **`MFPSB_HCTX`** value that is used as a handle to refer to a particular scan job.

To create a handle, provide the following:

- Printer IP address
- Profile name
- TCP port number

The function fails when it cannot communicate with the printer. If **`SinglePort`** is set to **`TRUE`**, then the port value is ignored.

Sample code

```
MFPSB_HCTX hCtx;
hCtx = MFPSB_CreateScanContext("myprinter.lexmark.com",
"mylabel", 9700);
if (hCtx == NULL)
{
   //There was an error creating the context.
}
```

Understanding the port range

The **`Set Port Range`** function designates the ports that Imaging Toolkit uses for receiving images. Lower and upper port range values are passed as a parameter, and must be set between 1024 and 65535. To disable the port range function and delete specified values, set the lower and upper port numbers to **`-1`**.

Sample code in Visual Basic

```
If isEmptyLowPort = True And isEmptyUpperPort = True Then
      //Disable the port range
        nLowerPort = -1
```

```
        nUpperPort = -1
        lError = MFPSB_SetPortRange(nLowerPort, nUpperPort)
        //Error code handling.
         .........
Else
      ..............
      //Setting the port range.
        nLowerPort = Val(txtLowerPort)
        nUpperPort = Val(txtUpperPort)
        lError = MFPSB_SetPortRange(nLowerPort, nUpperPort)
```

Sample code in C++

```
int lowerport = 1024;
int upperport = 65535;
DWORD dwd = MFPSB_SetPortRange(lowerport,upperport);
if(dwd == 0)
      //Success
else
      //Show error message
```

Note: If **`SinglePort`** is set to **`TRUE`**, then the port value is ignored.

Understanding scan settings specification

```
MFPSB_SetScanSetting()
```

Imaging Toolkit allows scan parameters for contexts. Each context has its own set of parameters. To query the parameters, use the **`MFPSB_QueryScanSetting()`** function.

Note: Calling the **`MFPSB_SetScanSetting()`** function updates one scan setting at a time. Call the function when updating a scan setting.

Sample code

```
dwError = MFPSB_SetScanSetting(hCtx, “RESOLUTION”, “200”);
if (dwError != MFPSB_E_SUCCESS)
{
     //Your code to handle this error
}
Char sValue[32];

Long sValueLen = 32;
  //Get the value for the resolution parameter

dwError = MFPSB_QueryScanSetting(hCtx, “RESOLUTION”, sValue,
&sValueLen);

if (dwError != MFPSB_E_SUCCESS)
{
     //Your code to handle the error
}
```

To retrieve a list of all scan settings for a specified context ID, use **`MFPSB_GetScanSettingsAsString`**.

Sample code

```
DWORD MFPSB_GetScanSettingsAsString(HANDLE hCtx, LPSTR sValue, LONG *sValueLen)
```

Where:

- **`hCtx [IN]`** is the handle or ID of the context.
- **`sValue [OUT]`** retrieves **`ScanSettings`** as a string delimited by Full Search and End of Transmission Block.
- **`sValueLen [OUT]`** is the number of context IDs.

Notes:

- If the function call is successful, then a zero value is returned. If the function call is not successful, then a nonzero value indicating the error code is returned. For more information on error codes, see "Possible API-related error messages" on page 23.
- Scan settings can be retrieved individually using **`MFPSB_QueryScanSetting`**, but this function should not be used to retrieve multiple scan settings.

Sample code for using the API to declare functions

```
Declare Function MFPSB_GetScanSettingsAsString Lib "mfpsb.dll"_
(ByVal hContext As Long,_
ByVal IpSettingValue As String,_
ByRef IpLength As Long) As Long
```

Sample code for using the function in Visual Basic

```
hCtx = IbContexts.Text
        IStringLength = 400

        IErrorCode = MFPSB_GetScanSettingsAsString(hCtx, sValue, IStringLength)
        FS = Chr(28)
        For I = 0 To 14
            nStartPos = 0
            nEqualPos = 0
            nFSPos = 0

            sSetting = sScanSetting(i)
            nStartPos = InStr(1, sValue, sSetting, vbTextCompare)
            If nStartPos > 0 The
                nEqualPos = nStartPos + Len(sSetting)
                nFSPos = InStr(Right(sValue, Len(sValue) - nEqualPos), FS)
                If I = 14 Then
                    sEdgeToEdgeVal = Mid$(sValue, nEqualPos +1, nFSPos - 1)
                    If sEdgeToEdgeVal = True Then
                        Check1.Value = 1
                    Else
                        Check.Value = 0
                    End If
                Else
                    tbScanSetting(i).Text = Mid$(sValue, nEqualPos + 1, nFSPos - 1)
                End If
            End If
```

Sample code for using the function in C++

```
CHAR value[10];
LONG valueLen = 10;
LONG* pValueLen = &vanlueLen;
dwd = MFPSB_GetScanSettingsAsString(contextid,value,pValueLen);
if(dwd==0)
{
      //Success
}
else
{
      //Show error message
}
```

Registering a callback function

```
MFPSB_RegisterCallback()
```

Note: This step is optional. However, if the non-blocking scan function is used, then we recommend using a callback function for Imaging Toolkit to receive status on the imaging process.

When registering the callback function, determine the events that must cause the Imaging Toolkit library to initiate the callback function. If a handle to a window is provided in the function, then the initialization of the callback occurs in the window procedure of this window. The imaging process runs on its own separate thread, and the calling application may require the callback to run in the same thread as the application. This process is the same for Visual Basic programs.

```
VOID_stdcall myCallback (MFPSB_HCTX hCtx, LONG lpId, LPVARIANT lpInfo)
{
   //Your own code representing what should be done
}
.
.
.
DWORD dwError;
dwError = MFPSB_RegisterCallBack(hCtx, (MFPSB_PFNCALLBACK)
myCallback, MFPSB_SBALLEVENTS, hWnd);
```

Calling the scan functions

```
MFPSB_StartScanBack()
MFPSB_StartScanBackNoBlock()
```

The two scan function options let you scan with or without a blocking function. The non-blocking versions allow the application to respond to your input. When you call the function, the profile is sent to the printer, and the Toolkit waits for the printer to connect back to it.

```
//Start the scan process.
dwError = MFPSB_StartScanBackNoBlock(hCtx);
if (dwError != MFPSB_E_SUCCESS)
{
      //The imaging process failed to start
      //handle the error
}
```

Uninitializing

```
MFPSB_Uninitialize()
```

This function releases all resources used by the library and deletes any remaining contexts.

Understanding scan context retrieval

We recommend calling these APIs when creating an instance of the application.

- **`MFPSB_GetInitContexts()`**—Determines the number of context IDs.

  Note: If the function call is successful, then a zero value is returned. If the function call is not successful, then a nonzero value indicating the error code is returned. For more information on error codes, see "Possible API-related error messages" on page 23.

- **`MFPSB_GetInitContextProfileInfo()`**—Retrieves the following details of each scan context and identified by a context ID:
  - Profile name
  - MFP IP address
  - Output directory
  - Output file name

  Note: If the function call is successful, then a zero value is returned. If the function call is not successful, then a nonzero value indicating the error code is returned. For more information on error codes, see "Possible API-related error messages" on page 23.

Sample code for using the APIs to declare functions in Visual Basic

```
Declare Function MFPSB_GetInitContexts Lib "mfpsb.dll"_
(ByRef address As Any,_
ByRef numContexts As Integer) As Long


Declare Function MFPSB_GetInitContextProfileInfo Lib "mfpsb.dll"_
(ByVal IContextID As Long,_
ByVal sProfileName As String,_
ByVal sMFPIP As String,_
ByVal sOutputFileDir As String,_
ByVal sOutputFileName as String,_
ByVal sIconFile as String) As Long
```

Sample code for using the functions in Visual Basic

```
sProfileName = Space$(35)
    sMFPIP = Space$(16)
    sOutputDir = Space$(225)
    sOutputFileName = Space$(35)
    sIconFile = Space$(260)
    IError = MFPSB_GetInitContexts(ByVal VarPtr(cList.ctx(oneIndex), sProfileName, sMFPIP,
sOutputFileDir, sOutputFileName, sIconFile)
        If IError = 0 Then
        IbContexts.AddItem (cList.ctx(oneIndex))
        Conexts(oneIndex, 0) = cList.ctx(oneIndex))
        Conexts(oneIndex, 1) = sMFPIP
        Conexts(oneIndex, 2) = sProfileName
        Conexts(oneIndex, 3) = sOutputFileDir
        Conexts(oneIndex, 4) = sOutputFileName
        Conexts(oneIndex, 5) = sIconFile
        numContexts = numContexts + 1
    End If
Next oneIndex
```

Sample code for using the functions in C++

```
LONG64 context[24];
MFPSB_HCTX initcontextPtr = context;
int numContexts = 24;
DWORD dwd = MFPSB_GetInitContexts(initcontextPtr, numContexts);
```

```
if(dwd == 0 )
{
      for(int iIndex=0 ; iIndex< numContexts; iIndex++)
      {
            //Show context ID
      }
}
else
{
      //Show error message
}
CHAR directory[32];
CHAR sIPAddress[32];
CHAR sOutputfilename[32];
CHAR sIconfile[32];
CHAR sProfileName[32];
dwd =
MFPSB_GetInitContextProfileInfo(contextid,sProfileName,sIPAddress,directory,sOutputfilename,s
Iconfile);
if(dwd == 0)
{
      //Success
}
else
      //Show error message
```

Using the sample application

We recommend using a C# application from the ***<install-dir>*\native\bin** folder, where ***<install-dir>*** is the installation directory of the application.

The sample application sends scan profiles to a printer, and then receives the scanned images. To facilitate the imaging process, the MFPSB API exposed by the DLL consists of 16 functions. You can run the sample application from the bin folder of this package.

To copy the sample to another location, store all DLL files from the bin folder in the application folder. You can also copy the sample in the PATH environment variable.

If an icon is used when creating profiles, then the style sheets contained in the bin folder must be in the following:

- PATH environment
- Application directory

1. From the application, create a context.
   a. Enter the IP address or host name of the printer.
   b. Type the profile name.
   c. Click **Create Context**.

   An integer appears in the Contexts list box. The default values for the scan settings also appear. Create Context calls the **`MFPSB_CreateCallback`** function to create the context, and then calls the **`MFPSB_RegisterCallbackFunction`** when successful.

   The callback function for this application writes information in the output window.

2. Set the port range.
   a. Set the lower and upper port range. The value can be from 1024 to 65535.
   b. Click **Set Port Range**.

   The port range is set by calling **`MFPSB_SetPortRange()`**.

   If port range settings are not specified, then the server attempts to open a socket on the first available port.

3 Enter the scan settings.

   a Enter a value for each scan setting.

   b Click **Set Values**.

   c To test the value, click **Get Values**.

   `MFPSB_SetScanSetting` is called for each setting.

   Get Values calls the **`MFPSB_QueryScanSetting`** for each setting and updates the fields with the result.

4 Select a directory and file name for the output.

   Type the name of the directory and the output file. If these fields are not specified, then the directory used is the one where the application is running. The file name matches the profile name. These names can be entered at any time before Do Scan is selected, and files are overwritten without prompting.

   If you select **Do Scan**, then the **`MFPSB_SetOutputFile`** API function is called before **`MFPSB_StartScanBackNoBlock`**.

5 Type the full path of the icon file.

   If the printer has a control panel with a welcome screen, then send an icon to the printer, and then show it on the welcome screen. If the Icon field is not empty when Do Scan is selected, then the **`MFPSB_SetWelcomeScreenIcon`** function is called.

   Note: Icon images can be GIF, JPEG, or PNG, with dimensions of 120 x 75 pixels and not exceeding 11KB.

6 Perform the scan.

   Clicking **Do Scan** sends a profile to the printer by calling the **`MFPSB_StartScanBackNoBlock`** API function. This function is a non-blocking call. Updates on status come through the callback function which shows text into the output window. The scan job is initiated when the shortcut number or icon at the printer is used.

Other available options and their functions

Setting	Description
Cancel Scan	Cancels the scan job
Exit	Closes the application
Remove	Deletes the profile currently selected in the Profile Name field
Release Context	Releases any resources used by the scan context
Open Scanned Images	Executes a shell command on the output file for opening the scanned image

Possible API-related error messages

Error message	Error number	What it means	Functions that can return the error
MFPSB_E_SUCCESS	0	A successful operation has completed.	N/A
MFPSB_E_BIND_FAILED	1	An error occurred when binding to a TCP port where the printer is connecting.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_TIMEOUT	3	The timeout value specified in **MFPSB_StartScanBack** has elapsed while waiting for a printer to connect.	**MFPSB_StartScanBack()**
MFPSB_E_INVALID_CONTEXT	4	An invalid **MFPSB_HCTX** value is passed to one of the API functions.	• **MFPSB_StartScanBack()** • **MFPSB_StartScanBackNoBlock()** • **MFPSB_RegisterCallback()** • **MFPSB_SetScanSetting()** • **MFPSB_QueryScanSetting()** • **MFPSB_SetOutputFile()** • **MFPSB_CancelScan()** • **MFPSB_ReleaseContext()** • **MFPSB_SetWelcomeScreenIcon()**
MFPSB_E_INVALID_SETTING_NAME	5	An invalid setting name is passed to **MFPSB_QueryScanSetting()** or **MFPSB_SetScanSetting()**.	• **MFPSB_QueryScanSetting()** • **MFPSB_SetScanSetting()**
MFPSB_E_INVALID_SETTING_VALUE	6	An invalid setting value is passed to **MFPSB_SetScanSetting()**.	**MFPSB_SetScanSetting()**
MFPSB_E_POINTER	8	A null pointer is passed as a parameter to a function that does not accept **NULL** for the argument.	• **MFPSB_RegisterCallback()** • **MFPSB_RemoveProfile()** • **MFPSB_SetWelcomeScreenIcon()**
MFPSB_E_CANCELLED	10	One of the StartScanBack functions failed because **MFPSB_CancelScan()** is called.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_UNKNOWN	13	The printer responded with invalid data when applying or removing the scan profile.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()** • **MFPSB_RemoveProfile()**

Error message	Error number	What it means	Functions that can return the error
MFPSB_E_CONNECTION_CLOSED	15	The printer closed the connection while sending scan data.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_CONNECT_FAILED	17	Imaging Toolkit is unable to connect to the printer.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_OPEN_OUTPUT_FILE_FAILED	18	Imaging Toolkit is unable to open the output file to save the scanned image.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_UNSUPPORTED_OP	19	The requested operation is unsupported. This error occurs when trying to set a welcome screen icon for a printer that does not have a control panel.	**MFPSB_SetWelcomeScreenIcon()**
MFPSB_E_COMMUNICATION	20	Corrupted data is received from the printer.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_BAD_FIELD	21	One of the profile fields for the printer is bad. This error usually occurs when a profile is given an existing name on the printer.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_MAX_STORAGE_REACHED	22	The printer cannot store any more profiles due to space limitations. Delete profiles from the printer.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_FUNCTION_DISABLED	23	The printer disabled the requested function.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_ALREADY_RUNNING	25	The **MFPSB_StartScanBackNoBlock** has already been called without completing the supplied context.	**MFPSB_StartScanBackNoBlock()**
MFPSB_E_HTTP_VERIFICATION_FAILURE	27	An error occurred when verifying the HTTP response preceding an attempt to upload a profile to the printer.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_JOB_OVERWRITTEN	28	The printer is at maximum capacity and discarded the profile, and then another profile was submitted.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**

Error message	Error number	What it means	Functions that can return the error
MFPSB_E_OPERATION_FAILED	29	A communication error occurred when applying the profile to the printer.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_SHORTCUT_UNAVAILABLE	30	There are no shortcuts left in the range [MinShortcut, MaxShortcut].	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_LOST_PROFILE	31	The profile sent to the printer is removed.	• **MFPSB_StartBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_INVALID_PORT_RANGE	32	• The lower port number is less than 1024. • The upper port number is greater than 65535. • The lower port number is greater than the upper port number.	• **MFPSB_StartScanBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_PORT_UNAVAILABLE	33	The server socket cannot be created because a port is not available.	• **MFPSB_StartScanBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_INVALIDDIR	34	The target directory does not exist, or the target file name is invalid.	• **MFPSB_StartScanBack()** • **MFPSB_StartScanBackNoBlock()**
MFPSB_E_INIT_CONTEXT_NOTFOUND	53	The context list is not found when Imaging Toolkit initialized.	• **MFPSB_GetInitContexts()** • **MFPSB_GetInitContextProfileInfo()**
MFPSB_E_PROFILE_NOTFOUND	54	The profile from the MFP is not found.	**MFPSB_RemoveProfile()**

Notices

Edition notice

January 2021

The following paragraph does not apply to any country where such provisions are inconsistent with local law: LEXMARK INTERNATIONAL, INC., PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in later editions. Improvements or changes in the products or the programs described may be made at any time.

References in this publication to products, programs, or services do not imply that the manufacturer intends to make these available in all countries in which it operates. Any reference to a product, program, or service is not intended to state or imply that only that product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any existing intellectual property right may be used instead. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by the manufacturer, are the user’s responsibility.

For Lexmark technical support, go to **http://support.lexmark.com**.

For information on Lexmark's privacy policy governing the use of this product, go to **www.lexmark.com/privacy**.

For information on supplies and downloads, go to **www.lexmark.com**.

GOVERNMENT END USERS

The Software Program and any related documentation are "Commercial Items," as that term is defined in 48 C.F.R. 2.101, "Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. 12.212 or 48 C.F.R. 227.7202, as applicable. Consistent with 48 C.F.R. 12.212 or 48 C.F.R. 227.7202-1 through 227.7207-4, as applicable, the Commercial Computer Software and Commercial Software Documentation are licensed to the U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein.

Trademarks

Lexmark and the Lexmark logo are trademarks or registered trademarks of Lexmark International, Inc. in the United States and/or other countries.

Firebird is a registered trademark of the Firebird Foundation.

All other trademarks are the property of their respective owners.

Index