Evolis SDK

DLL Technical specifications

Table of Contents

Introduction

Overview

This documentation describes how to easily use Evolis Premium Suite (supervised mode) through the associated library.

Package content :

- DLL and .lib Hle
- C++ demo
- C# Wrapper
- C# demo
- Documentation

Requirements:

- Evolis Premium Suite 2 installed (Standard or Supervision mode) and running.
- Evolis Premium Suite installed (Standard or Supervision mode) and running.
- Supervised Evolis printer available via ethernet or USB.
- Evolis API mode documentation.
- Evolis Firmware documentation.

EvoSdk1Base.dll and EvoSdk1Lib1.dll

Delivery is made of two libraries, EvoSdk1Base.dll and EvoSdk1Lib1.dll. The first one provides helpers for logging or encode/decode base64 data. The second one is the main library and is intended to communicate with the API mode by using plain C functions.

As a reminder, usually, the communication with Evolis card printers is made through the API mode. The developer has to open a connection (with named pipe or TCP socket) to the ESPF server then send requests, in the JSON format, to interact with.

Purpose of the EvoSdk1Lib1.dll library is to provide an overlay to easily connect and interact with the ESPF server. With those libraries you will not have to worry about managing the connection, building JSON requests, parsing the reply, etc...

You only have to create a HANDLE object to communicate with the ESPF server then create another HANDLE object for each printer you want to interact with. Once you have valid HANDLE object, just call the functions accordingly with what you want to do. There are more details later on the document.

It's important to keep in mind that this delivery is an overlay to the existing API mode. As a result, you will have to refer to the API mode documentation (ex: for minor and major status) and to the firmware documentation to be able to fully use the present libraries.

Keep in mind also that the printer model is used with the corresponding service and TCP port.

In the case of Primacy 2:

Configuration	ESP2
Printer name	Primacy 2
Service used	Evolis Premium Suite 2
Default PIPE value	Espf2Server00
TCP port	18200

In the case of Primacy:

Configuration	ESP1
Printer name	Primacy
Service used	Evolis Premium Suite
Default PIPE value	EspfServer00
TCP port	18000

C++ demo

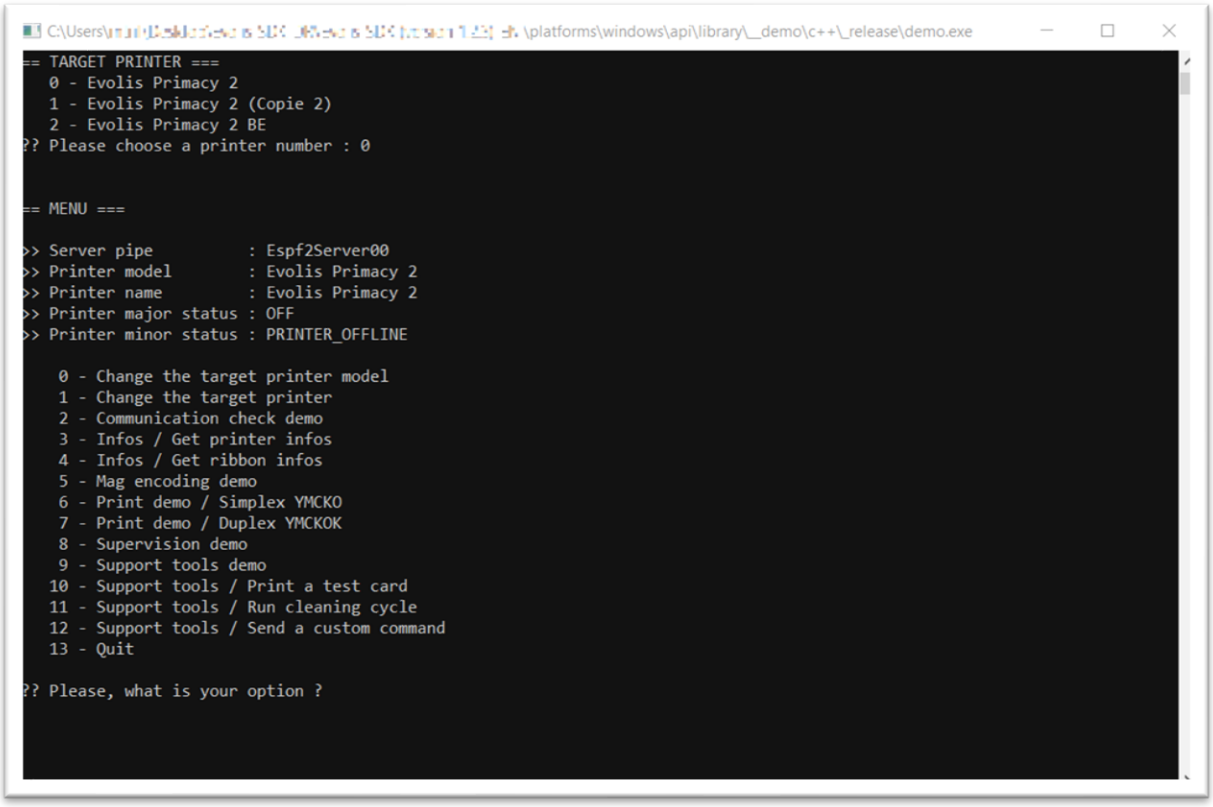
The directory DEMO_CPP contains a Visual Studio project with examples on how to use the present SDK. By default, the demo is run with the "Evolis Primacy 2" printer model name. You can override it by passing an argument to the executable or simply by updating the modelName variable in the main() function of the main.cpp file.

The libraries are provided for 32-bit and 64-bit Windows. At least 2 libraries and 4 headers are needed to build the project :

- EvoSdk1Base.dll: The DLL with logging and base64 functions. The library have 2 headers:
  - EvoSdk1Base.h: Contains functions declaration.
  - EvoSdk1BaseDef.h: Contains macros.

- EvoSdk1Lib1.dll: The main DLL containing the API described here. The library have 2 headers:
  - EvoSdk1Lib1.h: Contains functions declarations.
  - EvoSdk1Lib1Def.h: Contains enum types.

When launching the demo, a menu will ask you to choose the printer that you want to use. The demo is set by default to manage the Primacy 2 type printers:



To change the printer model, an argument needs to be sent from the command line as presented in the image below:

Please ensure that the correct service provider is installed and used according to the printer model, as shown here.

To launch a scenario simply enter the corresponding number then press enter key. Then you can refer to the source code to understand the process. Each scenario is stored in a corresponding Hle. For Example, the printing demo is stored in the demoPrint.cpp file, the mag encoding demo is stored in the demoMagEncoding.cpp file and so on...

One important thing to keep in mind when using the present SDK is that every string (char* type) returned by the EvoSdk1Lib1 library is managed by the library. The strings could be reused or destroyed by the library without you knowing it. That's why, if you want to later use the string, you have to backup it on your side. The advantage is that you never have to worry about releasing those strings returned by the library.
A simple example that illustrates it is in the demoPrint.cpp Hle on line 81. As you can see, we simply make a copy of the session id to be sure it stays valid for subsequent calls to the library.

C# demo

C# demo is the same as the C++ demo except that the DLL is used through a wrapper file named EVOSDK1Wrapper.cs. User must be careful about the charset when marshalling the pointer to avoid any error when retrieving data.

API details

This library provides features available in the ESPF and categorized per service. Each feature can be accessed by calling a set of methods as shown below.

Service Name	Purpose
GAPI	Functions to initialize/destroy handles.
PrinterSettings	To set the printer settings.
Print	To manage card printing.
Infos	To get information on the printer or it's ribbon.
MagEncoding	Read/write data on the card's magnetic tracks
Supervision	List printers and know theire status.
SupportTools	API for printer maintenance (test card, cleaning, ...).

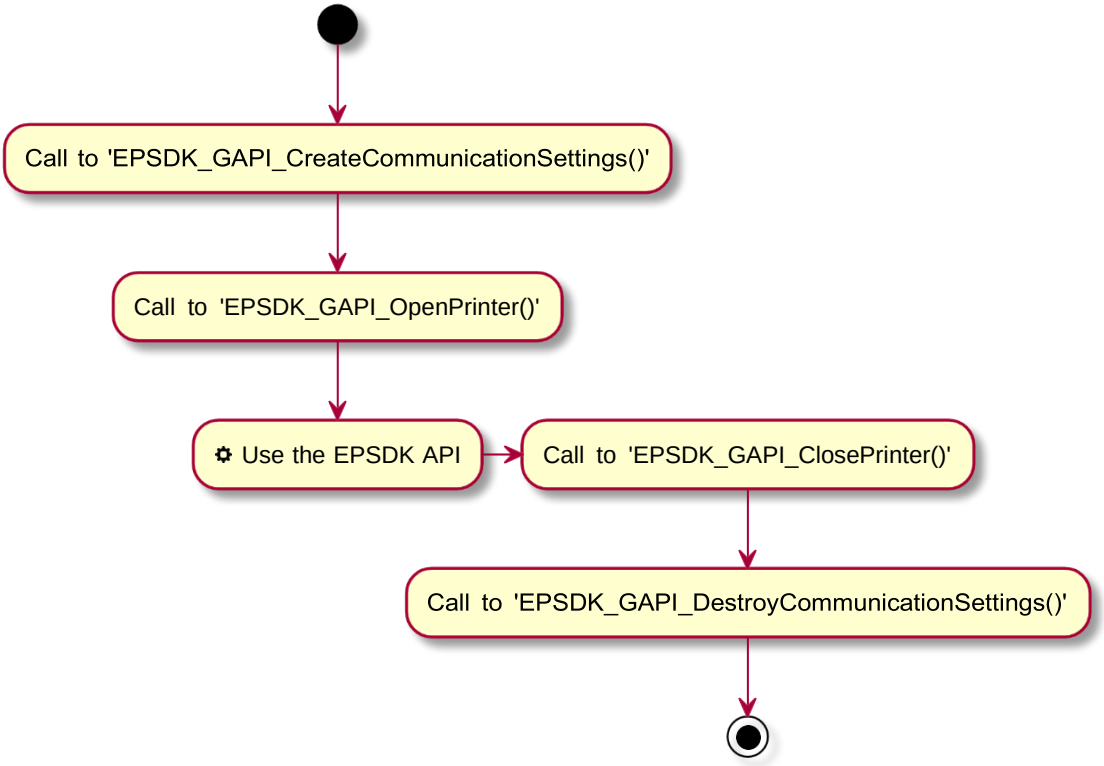
GAPI

Purpose :

- Create/destroy communication settings handle.
- Open/close printer connection handle.

The purpose of EPSDK_GAPI_*() functions is to manage the connection to the printer. To connect to the printer, we first have to connect to the ESPF server via the EPSDK_GAPI_CreateCommunicationSettings() method. The ESPF server act as a proxy between your code and the printer. Once the connection is ready you can create a new HANDLE to connect to the printer via the EPSDK_GAPI_OpenPrinter(). Of course, you have to release the HANDLEs when your job is done to avoid memory leaks.

The following diagram illustrates the explanation.



EPSDK_GAPI_CreateCommunicationSettings()

Create a HANDLE object that is used to communicate with the ESPF server. There is two way to connect to the server: with a named pipe or with a TCP socket. When connecting with named pipe, the function takes the pipe root and name as argument. When connecting via TCP, the function takes the IP and port as argument.

```
HANDLE EPSDK_GAPI_CreateCommunicationSettings(
  EPSDK_ConnectionType ct,  // `EPSDK_CT_Pipe` or `EPSDK_CT_IP`
  const char* param1,       // IP or pipe root, depending on the connection
  type const char* param2   // Port or pipe name, depending on the connection
  type
);
```

Return value :
Returns a valid communication settings HANDLE object on success or NULL otherwise.

EPSDK_GAPI_OpenPrinter()

Create a HANDLE object that is used to communicate with the printer. Warning, even if the function succeeds the printer is not necessarily valid. To be sure that the printer is valid, you must call EPSDK_Supervision_GetDeviceState() function.

```
HANDLE EPSDK_GAPI_OpenPrinter(
  const char* printerName,    // Printer name
  HANDLE comSettingHandle     // ESPF communication settings HANDLE
);
```

Return value :
Returns a printer HANDLE object when the function succeeds or NULL otherwise.

EPSDK_GAPI_DestroyCommunicationSetting()

To destroy a communication settings handle. Once the handle is destroyed, it's no longer possible to communicate with the server.

```
DWORD EPSDK_GAPI_DestroyCommunicationSettings(
  HANDLE comSettingHandle     // Communication settings `HANDLE` to destroy
);
```

Return value :
The function will return one of the following codes:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLEpassed in parameter.

EPSDK_GAPI_ClosePrinter

To destroy a printer handle. Once destroyed, it's no longer possible to use it to interact with the printer.

```
DWORD
  HANDLE                  //   Printer   `HANDLE`   to
);
```

Return value :
The function will return one of the following codes:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLEpassed in parameter.

Printer Settings

Purpose :

- Set the printer parameters

The EPSDK_PrinterSettings_*() function make use of the CMD.SendCommand service of the API modewhich means that the parameters that you set with those functions are directly sent to the printer.

EPSDK_PrinterSetting_SetCardInsertion()

This function allow user to set the card insertion mode. User must be aware that not all settings are available on each printer model. Please see the Pcim command of the firmware for more details.

```
int EPSDK_PrinterSettings_SetCardInsertion(
  HANDLE printerHandle,       // Printer `HANDLE` to communicate
  withEPSDK_InputTray value,  // See `EPSDK_InputTray` enum
  DWORD timeout               // Optional, command timeout
);
```

Return value :
The function will return one of the following codes:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLEpassed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_CD_CANT_GET_STATUS	Could happen when the printer is offline.
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_Printer_SetCardEjection()

The function allow user to set the card ejection mode. User must be aware that not all settings are available on each printer model. Please see the Pcem command of the firmware for more details.

```
int EPSDK_PrinterSettings_SetCardEjection(
  HANDLE printerHandle,       //  Printer  `HANDLE`  to  communicate
  withEPSDK_OutputTray value, // See `EPSDK_OutputTray` enum
  DWORD timeout               // Optional, command timeout
);
```

Return value
The function will return one of the following codes:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLEpassed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_CD_CANT_GET_STATUS	Could happen when the printer is offline.
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_PrinterSettings_SetCardEjectionOnError()

Function allows user to set the card ejection on error mode. User must be aware that not all settings are available on each printer model. Please see the Pcrm command of the firmware for more details.

```
int EPSDK_PrinterSettings_SetCardEjectionOnError(
  HANDLE printerHandle,       // Printer `HANDLE` to communicate with
  EPSDK_RejectBox value,      // See `EPSDK_RejectBox` enum
  DWORD timeout               // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_CD_CANT_GET_STATUS	Could happen when the printer is offline.
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_PrinterSettings_SetCardPreloading()

The function allows user to set the card pre-loading option on monochrome printing. Pre-loading optimizes cardflow production. Please see the Peoic command of the firmware for more details.

```
int EPSDK_PrinterSettings_SetCardPreloading(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  bool preloadCard,     // `True` to enable card preloading, `false` otherwise
  DWORD timeout         // Optional, command timeout
);
```

Return value
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_CD_CANT_GET_STATUS	Could happen when the printer is offline.
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_PrinterSettings_SetCardRetainingLength()

Only available on KC/KM model. Allow user to set the card retaining length in millimeter. Please see the Rleb command of the firmware for more details.

```
int EPSDK_PrinterSettings_SetCardRetainingLength(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  DWORD lengthValue,    // Length of card exceeding the bezel
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_CD_CANT_GET_STATUS	Could happen when the printer is offline.
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_INVALID_CMD_PARAM	The option is not supported.

EPSDK_PrinterSettings_SetCardRetainingDelayBeforeAction()

Only available on KC/KM model. Change the card retaining delay before the set action is done. User can set action to be done after delay using the method EPSDK_PrinterSettings_SetCardRetainingAction().

```
int    EPSDK_PrinterSettings_SetCardRetainingDelayBeforeAction(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  DWORD delayValue,     // Card retaining delay
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_CD_CANT_GET_STATUS	Could happen when the printer is offline.
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_INVALID_CMD_PARAM	The option is not supported.

EPSDK_PrinterSettings_SetCardRetainingAction()

Change the card retaining action after delay is out. User can set delay using the method EPSDK_PrinterSettings_SetCardDelayBeforeAction().

```
int EPSDK_PrinterSettings_SetCardRetainingAction(
  HANDLE printerHandle,     // Printer `HANDLE` to communicate with
  EPSDK_BezelAction value, // See `EPSDK_BezelAction` enum
  DWORD timeout             // Optional, command timeout
);
```

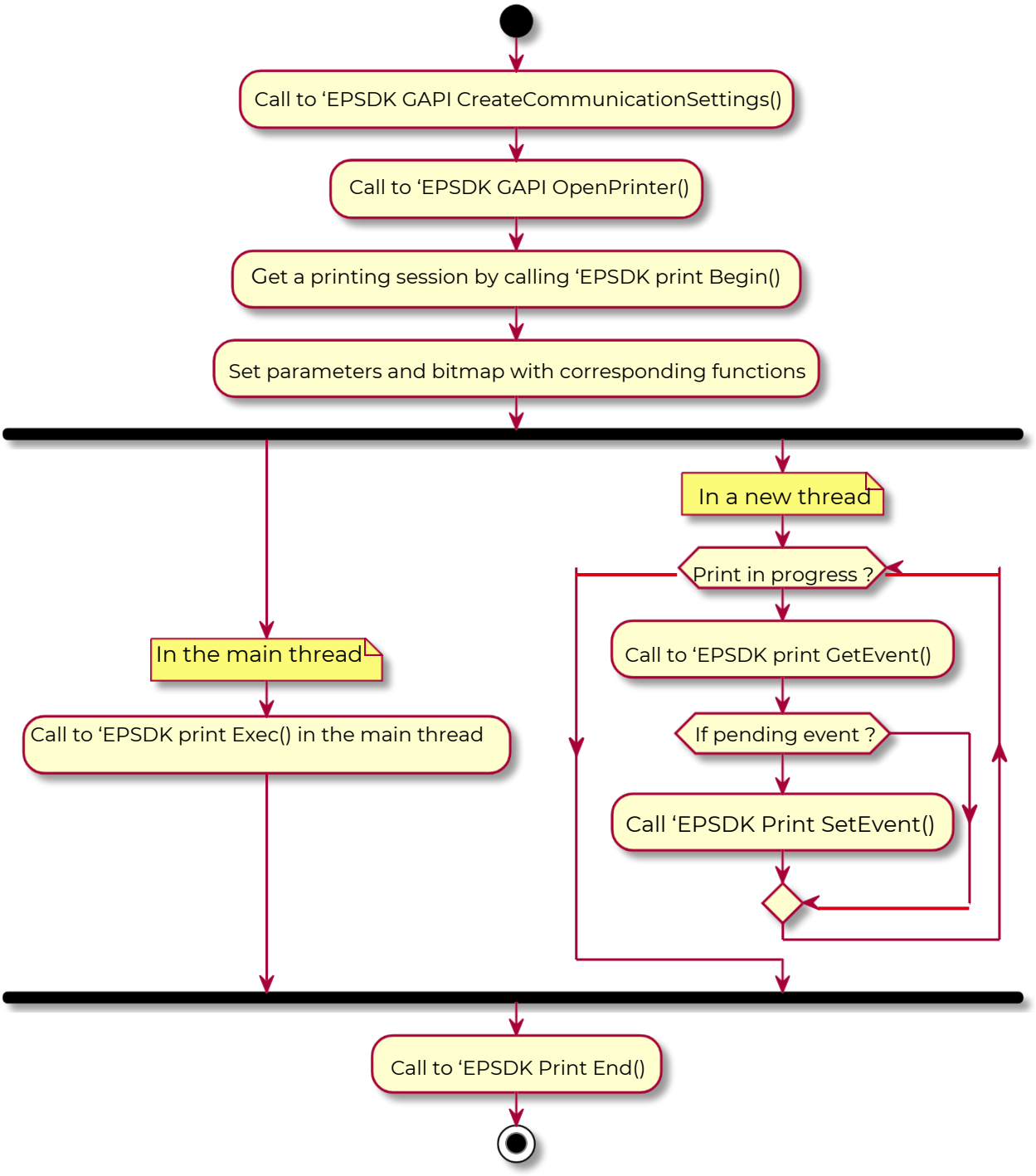
Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_CD_CANT_GET_STATUS	Could happen when the printer is offline.
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_INVALID_CMD_PARAM	The option is not supported.

Print

- Configure the printing jobs
- Print a card, whatever the printing system

The demoPrint() function in the DEMO_CPP project shows an example of how to print a card and manage events (feeder empty, cover open, etc.). In addition, the schema below describes how to develop a program that launch print sessions on an Evolis printer:



EPSDK_Print_Begin()

EPSDK_Print_Begin() create a new printing session. On success, the sessionId parameter is updated to contain a generated session id.

If a session is already registered for the printer, then the function fails and return with code EPSDK_PR_SESSION_ALREADY_RESERVED. A call to EPSDK_Print_GetJobID() will give you the id of the running session. You can ether close it or wait until the job is finished.

The sessionId parameter pointer is handled by the library. As a result of what the content could be overridden or destroyed by subsequent calls to the library. To be sure the value is still available you must clone the value in a variable that you own. Please see the demoPrint.cpp file in the demo project.

See the PRINT.Begin method of the API Mode for more details.

```
int EPSDK_Print_Begin(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  char** sessionId,        // On success, sessionId is updated to contain the session id
  const char* customId     // If not `NULL`, it's taken as the session id
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_PR_SESSION_ALREADY_RESERVED	A session is already reserved for the printer.

EPSDK_Print_Set()

EPSDK_Print_Set() allow user to set the printing parameters. See the PRINT.Set method of the API Mode for more details.

```
int EPSDK_Print_Set(
  HANDLE printerHandle,   // Printer `HANDLE` to communicate with
  const char** sessionId, // The id of the printing session
  const char*  data       // Printing parameters in the format "key1=value1;…;keyN=valueN"
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_PR_NO_PRINTING_SESSION	No printing session started for the printer.
EPSDK_PR_PRINT_ALREADY_IN_PROGRESS	Error, print already started.

EPSDK_Print_SetBitmap()

EPSDK_Print_SetBitmap() allow user to set bitmap data to print. See the PRINT.SetBitmap method of the API Mode for details.

The data parameter must be encoded with base64. The EvoSdk1Base_EncodeBinary() is here to help you transcode binary data of the bitmap to base64.

```
int EPSDK_PRINT_SetBitmap(
  HANDLE printerHandle,   // Printer `HANDLE` to communicate with
  EPSDK_FaceType face,    // Front or back face of the card, see `EPSDK_FaceType` enum
  EPSDK_PanelType panel,  // Ribbon choice, see `EPSDK_PanelType` enum
  const char** sessionId, // The id of the printing session
  const char * data       // Bitmap data encoded with base64
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_PR_NO_PRINTING_SESSION	No printing session started for the printer.
EPSDK_PR_PRINT_ALREADY_IN_PROGRESS	Error, print already started.

EPSDK_Print_Exec()

To start printing. First, you have to set parameters with EPSDK_Print_Set() and bitmap data with EPSDK_Print_SetBitmap(). This function is synchronous.

Before calling this function, you will have to create a new thread in charge of managing printer events. See EPSDK_Print_GetEvent() and EPSDK_Print_SetEvent() functions for details on how to manage events. See the PRINT.Printmethod of the API Mode for details.

```
int EPSDK_PRINT_Print(
  HANDLE printerHandle,   // Printer `HANDLE` to communicate with
  const char** sessionId, // The id of the printing session
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_PR_NO_PRINTING_SESSION	No printing session started for the printer.
EPSDK_PR_SETTING_OR_BITMAP_MISSING	Setting or bitmap missing, print aborted.
EPSDK_PR_PRINT_ERROR	Error while printing.
EPSDK_PR_PRINT_ALREADY_IN_PROGRESS	Error, print already started.
EPSDK_PR_PROCESSING_ERROR	Error in the driver while processing the data.
EPSDK_PR_JOB_CANCELED	Error, job was canceled.

EPSDK_Print_End()

Close the printing session. Be careful, if not call, the session stay opened and no one can open a session to start printing.
See the PRINT.End method of the API Mode for more details.

```
int EPSDK_PRINT_End(
  HANDLE printerHandle,   // Printer `HANDLE` to communicate with
  const char** sessionId, // The id of the printing session
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_PR_NO_PRINTING_SESSION	No printing session started for the printer.
EPSDK_PR_PRINT_ALREADY_IN_PROGRESS	Error, print already started.

EPSDK_Print_GetJobID()

To get the current printing session id. See the PRINT.GetJobId method of the API Mode for more details.

```
int EPSDK_Print_GetJobID(
  HANDLE printerHandle,   // Printer `HANDLE` to communicate with
  const char** sessionId, // The id of the printing session
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_PR_NO_PRINTING_SESSION	No printing session started for the printer.

EPSDK_Print_GetEvent()

Check for a pending event. An event is composed of a 'minorStatus' and associated event list. You have to call EPSDK_Print_GetEvent() to select the action that you want. See the SUPERVISION.GetEvent method of the Reference Guide for more details.

```
int EPSDK_PRINT_ERRMANAGEMENT_GetEvent(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  char** minorState,       // Will be filled in with the 'minorStatus' text
  char** actionList        // Will be filled in with actions in the format
'action1,action2,actionN'
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_PR_NO_PRINTING_SESSION	No printing session started for the printer.

A list of action will be displayed, the user needing to choose what action is appropriate according to the situation.

EPSDK_Print_SetEvent()

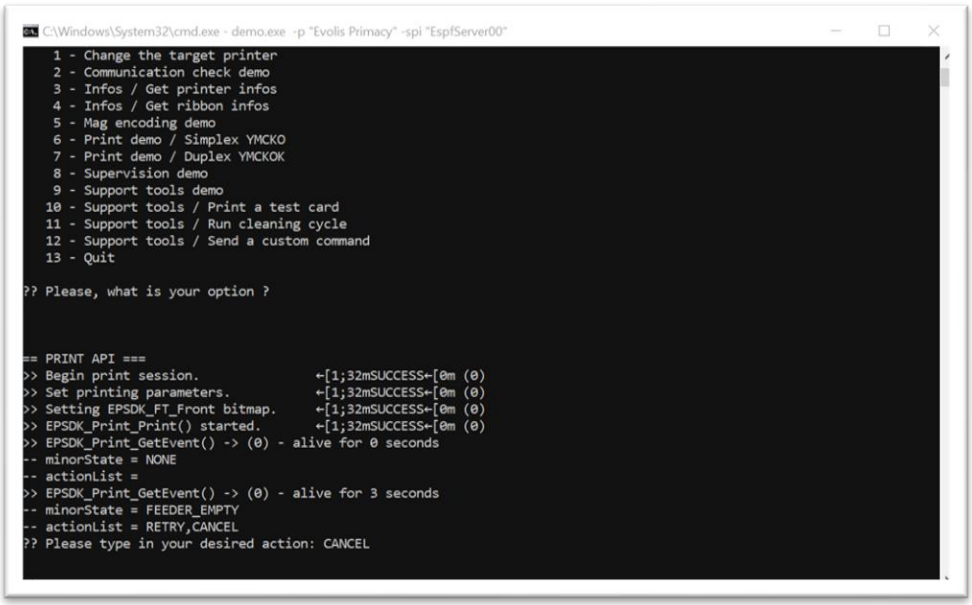
To select an action for a pending event. The action must be taken from the action list retrieve from a call to EPSDK_Print_GetEvent().
See the SUPERVISION.SetEvent method of the Reference Guide for more details.

```
int EPSDK_PRINT_ERRMANAGEMENT_SetEvent(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  const char* minorState, // The name of the pending event on which you want to set the action
  const char* action       // The name of the action you want to set on the event
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_SV_SET_ACTION_ERROR	Error when setting the action.
EPSDK_SV_NO_PENDING_EVENT	No pending event, nothing done.
EPSDK_SV_INVALID_ACTION	Invalid action for the pending event.
EPSDK_SV_INVALID_EVENT	The event is not the pending event.

You can select an action by typing it in the command line as shown in the image below:



Infos

- Retrieve printer information
- Retrieve ribbon information

EPSDK_Infos_GetPrinterInfos

EPSDK_Infos_GetPrinterInfos() allow user to retrieve printer information. See EPSDK_PrinterInfo description below for details.

```
int EPSDK_Infos_GetPrinterInfos(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  EPSDK_PrinterInfo info, // The printer info to get
  char** answer,           // The reply of the printer
  DWORD timeout            // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_Infos_GetRibbonInfos()

EPSDK_Infos_GetRibbonInfos() allow user to retrieve ribbon information. See EPSDK_RibbonInfo description below for details.

```
int EPSDK_Infos_GetRibbonInfos(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  RIBBON_INFO info,        // The ribbon info to get
  char** answer,           // The reply of the printer
  DWORD timeout            // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

MagEncoding

- Set specific parameters
- Read and write data on the card magnetic tracks

EPSDK_MagEncoding_ReadTrackData()

EPSDK_MagEncoding_ReadTrackData() allow user to read the specified magnetic track.

```
int EPSDK_MagEncoding_ReadTrackData(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  int trackNumber,         //  Magnetic  track  number  to  encode
  char** answer,           // The reply of the printer
  DWORD timeout            // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_ME_NOT_AVAILABLE	Magnetic encoding not available.
EPSDK_ME_BLANK_TRACK	The track is blank or unreadable.

EPSDK_MagEncoding_SetTrackData()

EPSDK_MagEncoding_SetTrackData() allow user to set the data to encode on the different magnetic tracks. User needs to remember about the different ISO settings on magnetic tracks (see magnetic ISO norm 7811/7813 for detailed information). See the Dm command of the firmware for details.

Note that the data is only set in the printer memory. The write is done when calling EPSDK_MagEncoding_WriteTrackData(), that's why there is no card movement made at this point.

```
int EPSDK_MagEncoding_SetTrackData(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  int trackNumber,         //  Magnetic  track  number  to  encode
  char** data,             // Data to write on the specified track
  DWORD timeout            // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_ME_NOT_AVAILABLE	Magnetic encoding not available.
EPSDK_ME_BAD_DATA_FORMAT	Invalid data format, see ISO norm.

EPSDK_MagEncoding_WriteTrackData()

EPSDK_MagEncoding_WriteTrackData() allow user to encode the data on all magnetic tracks.

All tracks' data must be previously set using the EPSDK_MagEncoding_SetTrackData() method. See the Smw command of the firmware for details.

```
int EPSDK_MagEncoding_WriteTrackData(
  HANDLE printerHandle,    // Printer `HANDLE` to communicate with
  DWORD timeout            // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_ME_NOT_AVAILABLE	Magnetic encoding not available.
EPSDK_ME_WRITE_ERROR	Error on writing data to the magnetic tracks.

EPSDK_MagEncoding_SetCOERCIVITY()

EPSDK_MagEncoding_SetCOERCIVITY() allow user to set the coercivity for magnetic encoding. See the Pmc command of the firmware for details.

The coercivity is implied by the card, either EPSDK_CO_Hico or EPSDK_CO_Loco. There is also the EPSDK_CO_Auto which tries to detect the card coercivity.

```
int EPSDK_MagEncoding_SetCOERCIVITY(
  HANDLE printerHandle,          // Printer `HANDLE` to communicate
  withEPSDK_Coercivity coercivity, // Coercivity of the card
  DWORD timeout                  // Optional, command timeout
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_ME_NOT_AVAILABLE	Magnetic encoding not available.

EPSDK_MagEncoding_SetIsoNorm()

EPSDK_MagEncoding_SetIsoNorm() allow user to set the ISO magnetic norm on the different tracks. See the Pmt command of the firmware for details.

```
int EPSDK_MagEncoding_SetIsoNorm(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  DWORD trackNumber,    //  Magnetic  track  number  to  encode
  EPSDK_IsoNorm iso     // The iso norm to set
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_ME_NOT_AVAILABLE	Magnetic encoding not available.

Supervision

- Lists available Evolis printers
- Retrieve the device state
- Enable/disable supervision on the device

EPSDK_Supervision_EnumEvolisSupervisedPrinters()

EPSDK_Supervision_EnumEvolisSupervisedPrinters() allow user to get the list of Evolis supervised printers. There is three level of details :
level = 0: Printer name only.
level = 1: Printer name + major status.
level = 2: Printer name + full status.

On success, the result answer will contain a string with printers separated with ;. Printer name and states are separated with ,. See SUPERVISION.List method of the API Mode for details.

```
DWORD EPSDK_Supervision_EnumEvolisSupervisedPrinters(
  HANDLE comHandle,      // An HANDLE allocated by EPSDK_GAPI_CreateCommunicationSettings()
  int level,             // A number from 0 to 2
  const char* modelName, // The model name of the printers you want to list
  char** answer,         // On success, contains the printer list
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_Supervision_GetDeviceState()

EPSDK_Supervision_GetDeviceState() allow user to retrieve the device state. See SUPERVISION.GetState method of the API Mode for details and full list of expected states.

```
DWORD EPSDK_Supervision_GetDeviceState(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  char** majorStatus, // On success, contain the major status
  char** minorStatus  // On success, contain the minor status
);
```

Return value :

When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.

EPSDK_Supervision_AddDevice()

EPSDK_Supervision_AddDevice() allow user to activate supervision on the device.

```
DWORD EPSDK_Supervision_AddDevice(
    HANDLE printerHandle, // Printer `HANDLE` to communicate with
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_SV_DEVICE_ALREADY_SUPERVISED	The printer is already supervised
EPSDK_INTERNAL_ERROR	Unknown internal error.

EPSDK_Supervision_RemoveDevice()

EPSDK_Supervision_RemoveDevice() allow user to disable supervision on the device.

```
DWORD EPSDK_Supervision_RemoveDevice(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_SV_DEVICE_ALREADY_UNSUPERVISED	The printer is already un-supervised
EPSDK_INTERNAL_ERROR	Unknown internal error.

SupportTools

- Send specific command(s)
- Reset communication if broken
- Retrieve binary status (during and out of printing task)
- Print test card
- Clean printer
- Card movement out of the printing task

EPSDK_SupportTools_SendCommand()

EPSDK_SupportTools_SendCommand() allow user to send a command to the printer. See the firmware documentation for available commands and parameters. The function uses the CMD.SendCommand of the API Mode.

```
int EPSDK_SupportTools_SendCommand(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  const char* command, // Command and params to execute
  char** answer,        // Optional, the printer answer
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_SupportTools_ResetCommunication()

EPSDK_SupportTools_ResetCommunication() allow user to reset communication between the ESPF server and the printer. It can be used in some case where there is unexpected errors and you can't do anything to regain control of the printer. See the CMD.ResetCom method of the API Mode for details.

```
int EPSDK_SupportTools_ResetCommunication(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_SupportTools_GetDeviceBinaryStatus()

EPSDK_SupportTools_GetDeviceBinaryStatus() allow user to retrieve the binary status of the specified printer. The result string is base64 encoded data. It can be decoded to a byte array with EvoSdk1Base_DecodeBinary(). See the API Mode doc for details on how to use decoded data.

```
int EPSDK_SupportTools_GetDeviceBinaryStatus(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  char** b64Status,     // On success, will contain the base64 encoded status
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.

EPSDK_SupportTools_PrintTestCard()

EPSDK_SupportTools_PrintTestCard() allow user to print the test card. See the St command of the firmware for details.

```
int EPSDK_SupportTools_PrintTestCard(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  char** errorString,   // Optional, could contain printer error
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_SEE_ERROR_STRING	The errorString param is filled in with printer error.

EPSDK_SupportTools_RunCleaningCycle()

EPSDK_SupportTools_RunCleaningCycle() allow user to run a cleaning cycle. See the Scp command of the firmware for details.

```
int EPSDK_SupportTools_RunCleaningCycle(
  HANDLE printerHandle, // Printer `HANDLE` to communicate with
  char** errorString,   // Optional, could contain printer error
  DWORD timeout         // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_SEE_ERROR_STRING	The errorString param is filled in with printer error.

EPSDK_SupportTools_MoveCard()

EPSDK_SupportTools_MoveCard()allow user to move a card to different spot in the printer.

```
int EPSDK_SupportTools_MoveCard(
  HANDLE printerHandle,     // Printer `HANDLE` to communicate with
  EPSDK_CardDestination cd, // See `EPSDK_CardDestination` enum
  DWORD timeout             // Optional, command timeout
);
```

Return value :
When the function is successfully completed, status below will be returned:

Return code name	Description
EPSDK_NO_ERROR	Success
EPSDK_INVALID_HANDLE	Invalid HANDLE passed in parameter.
EPSDK_INVALID_PARAMETER	One of the parameter is invalid.
EPSDK_ERR_NO_COMMUNICATION_WITH_ESPF_SERVICE	No communication with ESPF server
EPSDK_ERR_NO_COMMUNICATION_WITH_PRINTER	No communication with printer
EPSDK_INTERNAL_ERROR	Unknown internal error.
EPSDK_TIMEOUT	Request timeout.
EPSDK_SEE_ERROR_STRING	The errorString param is filled in with printer error.

Enumerations

```
enum EPSDK_ConnectionType {
    EPSDK_CT_Unknown,
    EPSDK_CT_Pipe,
    EPSDK_CT_IP,
};
```

```
enum EPSDK_FaceType {
    EPSDK_FT_Front,
    EPSDK_FT_Back,
};
```

```
enum EPSDK_PanelType {
    EPSDK_PT_Color,
    EPSDK_PT_Resin,
    EPSDK_PT_Varnish,
};
```

```
enum    EPSDK_IsoNorm    {
    EPSDK_IN_Iso1 = 1,
    EPSDK_IN_Iso2 = 2,
    EPSDK_IN_Iso3 = 3,
};
```

```
enum  EPSDK_Coercivity  {
    EPSDK_CO_Hico = 'h',
    EPSDK_CO_Loco = 'l',
    EPSDK_CO_Auto = 'a',
};
```

```
enum EPSDK_CardDestination {
    EPSDK_CD_PrintPos,        // Fw cmd: Si - Default value
    EPSDK_CD_BackPos,         //      Fw      cmd:      Sib
    EPSDK_CD_SmartPos,        //      Fw      cmd:      Sis
    EPSDK_CD_ContactLessPos,     //     Fw     cmd:     Sic
    EPSDK_CD_EjectCard,       //       Fw       cmd:       Se
    EPSDK_CD_RejectCard,      // Fw cmd: Ser
    EPSDK_CD_InsertEjectCard, // Fw cmd: Sie - Insert then eject the card
};
```

```
enum EPSDK_InputTray {
    EPSDK_IT_Feeder,          // Fw cmd: Pcim;F
    EPSDK_IT_Auto,            // Fw cmd: Pcim;B
    EPSDK_IT_Manual,          // Fw cmd: Pcim;M - Not available on KC model
    EPSDK_IT_Bezel,           // Fw cmd: Pcim;I - Only available on KC model
};
```

```
enum EPSDK_OutputTray {
    EPSDK_OT_Hopper,          // Fw cmd: Pcem;D
    EPSDK_OT_Rear,            // Fw cmd: Pcem;R
    EPSDK_OT_Bezel,           // Fw cmd: Pcem;I
    EPSDK_OT_Manual,          // Fw cmd: Pcem;M
    EPSDK_OT_LowerSlot,       // Fw cmd: Pcem;K
};
```

```
enum EPSDK_RejectBox {
    EPSDK_RB_Default,         // Fw cmd: Pcrm;D
    EPSDK_RB_Hopper,          // Fw cmd: Pcrm;R
    EPSDK_RB_Manual,          // Fw cmd: Pcrm;M
    EPSDK_RB_Bezel,           // Fw cmd: Pcrm;I
    EPSDK_RB_LowerSlot,       // Fw cmd: Pcrm;K
    EPSDK_RB_Locked,          // Fw cmd: Pcrm;L
};
```

```
enum EPSDK_BezelAction {
    EPSDK_BA_Rejected,        // Fw cmd: Pbc;A;N
    EPSDK_BA_Inserted,        // Fw cmd: Pbc;A;R
    EPSDK_BA_Nothing,         // Fw cmd: Pbc;A;D
};
```

```
enum EPSDK_RibbonInfo {
    EPSDK_RI_Type,              // Fw cmd: Rrt;type
    EPSDK_RI_Zone,              // Fw cmd: Rrt;zone
    EPSDK_RI_RemainingCapacity, // Fw cmd: Rrt;count
    EPSDK_RI_TotalCapacity,     //  Fw  cmd:  Rrt;qty
    EPSDK_RI_BatchNumber,       //  Fw  cmd:  Rrt;k7
    EPSDK_RI_ProductCode,       //  Fw  cmd:  Rrt;ref
    EPSDK_RI_ManufacturedDate,  //  Fw  cmd:  Rrt;date
    EPSDK_RI_Description,       // Fw cmd: Rrt;label
};
```

```
enum EPSDK_PrinterInfo {
    EPSDK_PI_Fw,                    // Fw  cmd: Rfv
    EPSDK_PI_Name,                  // Fw  cmd: Rsn
    EPSDK_PI_Zone,                  // Fw  cmd: Rzp
    EPSDK_PI_Model,                 // Fw  cmd: Rtp
    EPSDK_PI_Duplex,                // Fw  cmd: Rflo
    EPSDK_PI_SmartOffset,           // Fw  cmd: Ros
    EPSDK_PI_RetainingLength,       // Fw  cmd: Rleb
    EPSDK_PI_RetainingDelay,        // Fw  cmd: Rbc;A
    EPSDK_PI_RetainingAction,       // Fw  cmd: Rbc;D
    EPSDK_PI_PrintedPanelsCount     // Fw  cmd: Rco;gp
    EPSDK_PI_InsertedCardsCount,    // Fw  cmd: Rco;gc
    EPSDK_PI_OptionEthernet,        // Fw  cmd: Rtp;E
    EPSDK_PI_OptionWifi,            // Fw  cmd: Rtp;W
    EPSDK_PI_OptionMag,             // Fw  cmd: Rtp;M
    EPSDK_PI_OptionSmart,           // Fw  cmd: Rtp;S
    EPSDK_PI_OptionContactLess,     // Fw  cmd: Rtp;C
    EPSDK_PI_OptionFlip,            // Fw  cmd: Rtp;F
    EPSDK_PI_OptionLaminator,       // Fw  cmd: Rtp;L
};
```

General information

All functions (except EPSDK_GAPI_*() ones) return 0 on success. Other return codes are referenced in the EPSDK_ReturnCode enum.

Some functions take a char** parameter which is updated to point to an internal memory. The returned pointer could be overridden or destroyed by subsequent calls to the library. To be sure the value is still available you have to clone the value in a variable that you own. For example :

```
char* reply;
DWORD ec;

if (!(ec = EPSDK_Print_Begin(ph, &reply))) {
    char sessionId[128];

    strcpy(sessionId, reply); // save reply locally
    //...
    EPSDK_Print_End(ph, sessionId);
}
```

DISCLAIMER

While Evolis makes every effort to deliver high quality products, we do not guarantee that our products are free from defects. Our SDK, samples and demo software, any content or documentation delivered in this package (Evolis SDK) is provided "as is". The use of it is at your own risk.

Evolis makes no warranties as to performance, merchantability, fitness for a particular purpose, or any other warranties whether expressed or implied.

No oral or written communication from or information provided by Evolis shall create a warranty.

Under no circumstances shall Evolis be liable for direct, indirect, special, incidental, or consequential damages resulting from the use, misuse, or inability to use this Software Development Kit (named Evolis SDK), even if Evolis has been advised of the possibility of such damages.

Evolis - 14 Avenue de la Fontaine - ZI Angers-Beaucouzé 49070 Beaucouzé - France
T +33 (0) 241 367 606 - F +33 (0) 241 367 612 - **info@evolis.com**

www.evolis.com