Software Integration Guide

Topaz SigPlusExtLite SDK

Designed for use in Chrome, Firefox, Opera, and Edge
Browser Extension Frameworks

Version 1.0 R1013
Last Update: January 3, 2018

Table of Contents

1.0 – Introduction

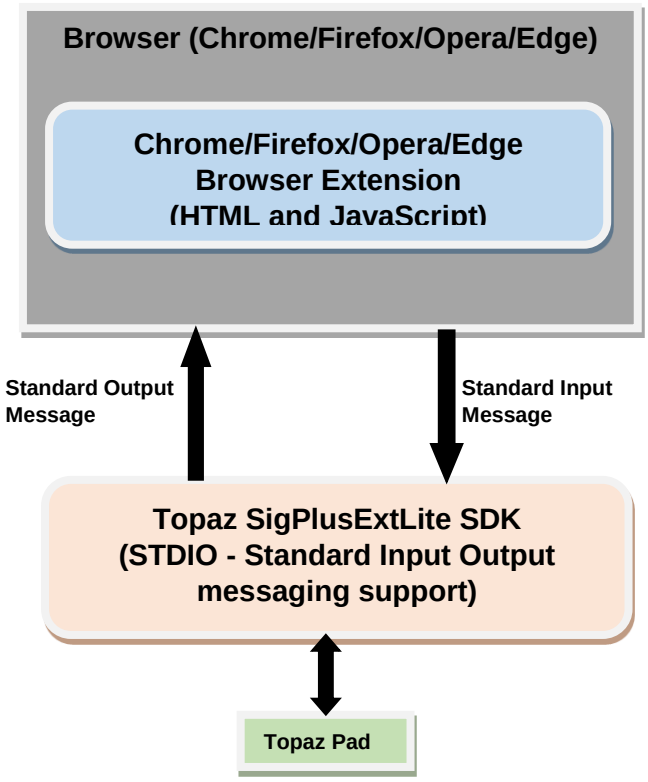
The Topaz SigPlusExtLite SDK offers a mechanism and platform for developers and integrators to capture handwritten signatures securely for web applications running in the Chrome, Firefox, Opera, and Edge browsers. The SDK provides capabilities for capturing biometric handwritten signatures using electronic signature pads from Topaz Systems.

The SDK exports the images of the captured signatures, as well as the raw biometric signature data in different formats. The images can be used in any application requiring signature images.

2.0 – Overview and Architecture

As Google Chrome and Mozilla Firefox are discontinuing support for plug-ins running inside the browsers, neither Java applets nor NPAPI plug-ins will be able to be used with browsers in the future. The SigPlusExtLite browser extensions are designed to provide web pages with the capability to capture signatures using Topaz signature pads connected to client desktops.

The diagram below shows the high-level overview of the solution with critical components involved.



2.1 – Topaz SigPlusExtLite SDK

The Topaz SigPlusExtLite SDK has been developed as a standard C#.Net application. It has built–in mechanism to capture signatures using Topaz signature capture devices and also to expose the raw biometric signature data in Topaz SigString format and the signature as an image (JPG, PNG, etc.). Its interfaces are implemented as Standard Input and Output streams under the Chrome, Firefox, Opera, and Edge Browser Extension frameworks. The SDK processes the input text messages from the Chrome, Firefox, Opera, and Edge browsers and executes the requests asynchronously, and when a task is complete sends back the status or output data as an output text message. It will host all the User Interface functions for capture and display of signatures from devices.

Browsers run this SDK in a separate process, launch it through Google Connect APIs, and send a notification back to the Extension when the application is ended by the user.

2.2 – Chrome Extension/Webpage

Chrome Extensions are the HTML, JavaScript, and CSS based code modules that are launched during startup of the browser or launched on demand from web page JavaScript. The extensions use JavaScript based Google Native Messaging APIs to launch and communicate with the Topaz SigPlusExtLite SDK for signature capture and other relevant features. The extension listens for the output messages from SigPlusExtLite and processes them accordingly. Google Native Messaging has a Connect API to launch the applications (which can process the standard input and output messages) in this SigPlusExtLite SDK and a Disconnect event to let the web page know about termination of the native host application. Using Connect and Disconnect, the life cycle of the native host application can be controlled. Also, Google Native Messaging APIs have mechanisms to send input messages to the SigPlusExtLite SDK and receive output messages from applications.

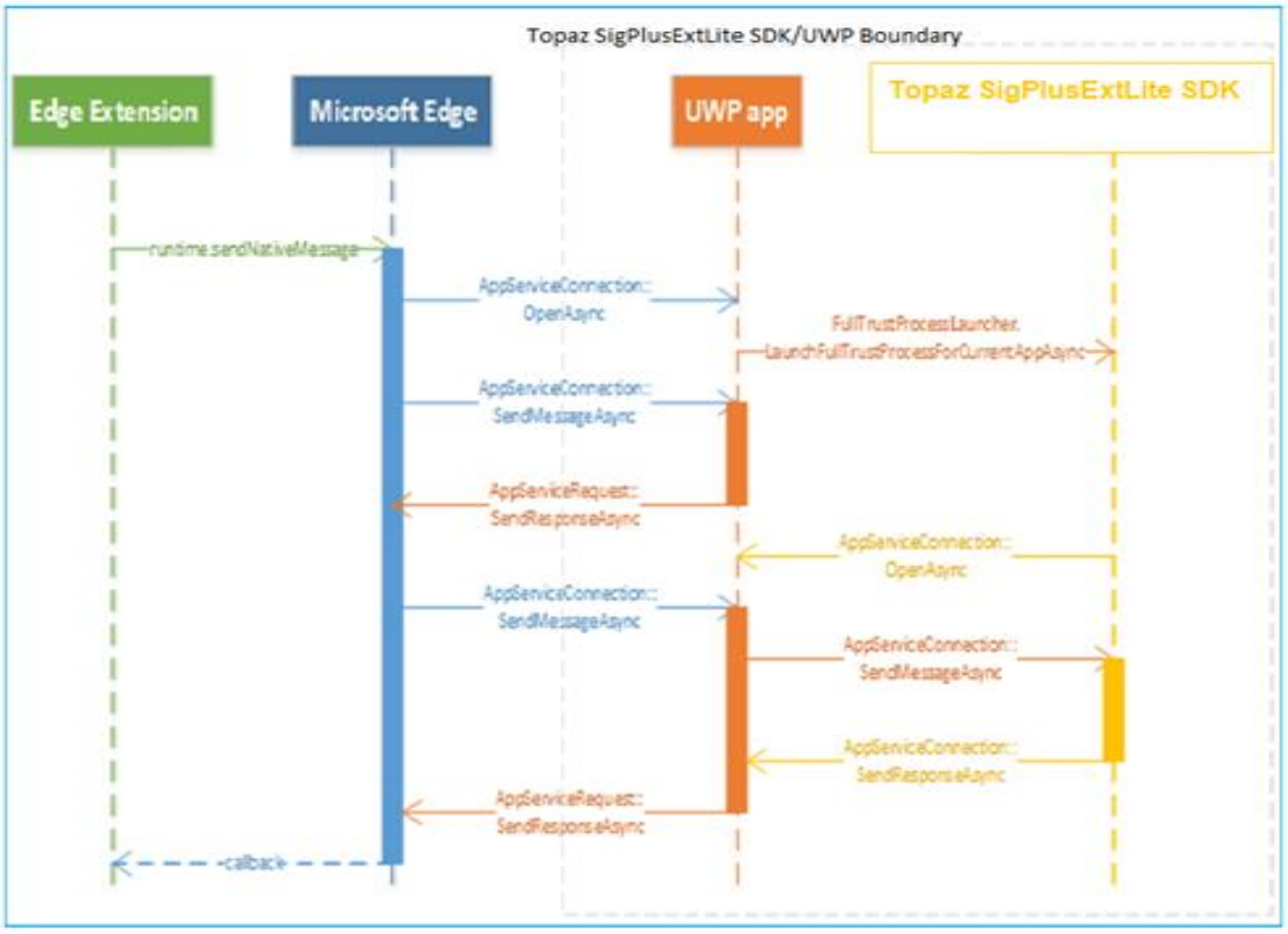
The Opera browser supports installation and usage of Chrome extensions. Topaz SigPlusExtLite SDK leverages the same to capture signatures within Opera browsers.

2.3 – Firefox Extension/Webpage

Firefox Extensions are the HTML, JavaScript, and CSS based code modules that are launched during startup of the browser. The Firefox extension uses Chrome Native Messaging APIs to launch the Topaz SigPlusExtLite SDK and send and receive input and output messages. It also has the required callback mechanism to notify the calling module when the launched application is terminated or when some error has occurred.

2.4 – Edge Extension/Webpage

Edge Extensions are the HTML, JavaScript, and CSS based code modules that are launched during startup of the browser. The Edge extension uses the AppService API to launch the Topaz SigPlusExtLite SDK and to send and receive input and output messages. UWP apps use the AppService API to communicate with one another. Because of this, Microsoft Edge extensions can communicate with UWP apps. The Microsoft Edge native messaging extension has both a companion UWP app and a Topaz SigPlusExtLite SDK as shown in the diagram below.



3.0 – Key Features

The Topaz SigPlusExtLite SDK provides the following features:

- Set the minimum number of signature points required to qualify a signature as valid
- Initiate signature capture
- Clear the captured signature
- Get information about the connected signature pad
- Export a signature image as PNG/JPG (base64 encoded string)
- Export the raw biometric signature data in encrypted base 64 string format.
- Export the raw signature data in a Compressed SigString format.
- Export signature images with custom text drawn beside the handwritten squiggle.

4.0 – Operating Systems Supported

The Topaz SigPlusExtLite SDK can be integrated into web pages running in the latest versions of Chrome, Firefox, and Opera browsers installed on Windows 7/8/8.1/10 32-bit operating systems. For 64-bit Windows operating systems running the 64-bit Chrome browser, the SDK should be run as a 32-bit application. For Edge, the Topaz SigPlusExtLite SDK can be integrated into web pages running in Edge browser installed on the Windows 10 operating system.

The samples have been tested in the latest version of Chrome, Firefox, Opera, and Edge browsers. Hence, it is recommended that you install the latest version of the Chrome (59 or higher), Firefox (50 or above), Opera (46 or above), and Microsoft Edge browsers in Windows 10.

Note: Microsoft Visual Studio 2008 with .NET Framework 3.5 is the development IDE used in developing the SigPlusExtLite SDK, and hence .NET Framework 3.5 should be available in the end user Windows computer.

Note: Microsoft Visual Studio 2015 with .NET Framework 4.5 is the development IDE used in developing the Edge Extension, and hence .NET Framework 4.5 should be available in the end user Windows 10 computer.

5.0 – Signature Capture Devices

The SigPlusExtLite SDK supports capturing signatures using electronic signature pads from Topaz Systems.

6.0 – Installing and Running SigPlusExtLite

For step-by-step instructions, view the SigPlusExtLite User Installation Guide at: www.topazsystems.com/Software/SigPlusExtLite_UserInstall.pdf. For information on end user deployment, view Section 9.0 of this guide.

7.0 – SigPlusExtLite Browser Integration for Signature Capture

For web pages running in Chrome, Firefox, Opera and Edge, the only required step is to raise and listen for predefined custom HTML events within the web page.

7.1 – Launching the Extensions from a Webpage

The first step is to detect if SigPlusExtLite extension has been installed in the browser. The following code snippet demonstrates how to detect if the SigPlusExtLite extension is not installed or disabled in the browser. If not installed it displays an alert.

```
var isInstalled = document.documentElement.getAttribute('SigPlusExtLiteExtension-installed');
if (!isInstalled) {
        alert("SigPlusExtLite extension is either not installed or disabled. Please install or enable extension.");
        return
}
```

The SigPlusExtLite extension relies on custom HTML events for communication between the web page and the extensions and vice versa.

The Chrome, Firefox, Opera, and Edge Extensions load during browser start up and register a custom HTML event named “SignStartEvent”. Web pages wishing to capture the signature using the SigPlusExtLite SDK in Chrome, Firefox, Opera, and Edge browsers have to raise the custom HTML event “SignStartEvent” and send an input message to the SDK as an event attribute.

Once the signature capture task is completed, the SigPlusExtLite extensions raise a custom HTML event named “SignResponse” and pass the output message as an event attribute. Web pages should register and implement the “SignResponse” event for processing the output from the extension.

The following code snippet demonstrates raising the custom HTML event “SignStartEvent” to initiate signature capture and also register and implement the “SignResponse” event for processing the output from the SDK.

```
var message = { "firstName": "", "lastName": "", "eMail": "", "location": "", "imageFormat": 1, "imageX": imgWidth,
"imageY": imgHeight, "imageTransparency": false, "imageScaling": false, "maxUpScalePercent": 0.0,
"rawDataFormat": "ENC", "minSigPoints": 25 };

                top.document.addEventListener('SignResponse', SignResponse, false);
                var messageData = JSON.stringify(message);
                var element = document.createElement("MyExtensionDataElement");
                element.setAttribute("messageAttribute", messageData);
                document.documentElement.appendChild(element);
                var evt = document.createEvent("Events");
                evt.initEvent("SignStartEvent", true, false);
                element.dispatchEvent(evt);


function SignResponse(event)
{
        var str = event.target.getAttribute("msgAttribute");
        var obj = JSON.parse(str);
        //Process the response
}
```

8.0 – Signature Capture and Data Export

As the SigPlusExtLite SDK is designed to support Standard Input and Output streams for communication between the browser extensions and the SDK, only text data can be exchanged between the applications. The Input message triggers signature capture, and the input message itself contains all the required data as payload.

The Output message payload will contain the signature signed status, the image data (in the specified format), the raw signature data in base 64 format, connected signature pad information, and a parameter to carry the error message in case signing fails.

The Chrome Google native messaging API (through which the Extension sends and receives data to the SDK) mandates the text data to be in JSON format, hence the input and output messages should be in JSON format and for convenience the same JSON format is followed for Firefox as well.

The format of the JSON message is

{text: value1, text1: value2}

where ‘text’ and ‘text1’ are the names of the JSON parameters.

8.1 – Signature Capture

The Topaz SigPlusExtLite SDK supports only one input message which will trigger the signature capture using the Topaz signature pads.

It gathers all the required inputs (like signature information (First Name, Last Name, Email, Location), the signature image and raw data format (Image: JPG/PNG, RawData: ENC), width and height of the image, background image transparency, etc.) for export after the capture.

8.1.1 – INPUT Message

Here is a sample JSON string containing all the supported INPUT.

```
{"firstName": "John" , "lastName": "Doe" , "eMail": "johndoe@demo.com" , "location": "LosAngels,CA" ,
"imageFormat": 2 , "imageX": 300 , "imageY": 150 , "imageTransparency": true , "imageScaling": false,
"maxUpScalePercent": 0.0, "rawDataFormat": "ENC" ,  "minSigPoints":25};
```

Parameter	Description
firstName	Signature Detail First Name as String.
lastName	Signature Detail Last Name as String.
eMail	Signature Detail Email as String.
location	The signing location as String, could be a physical location or longitude and latitude.
imageFormat	Format of the signature image to be exported after signature capture. Send 1 for JPG and 2 for PNG. The default format is PNG.
imageX	Physical width of the signature image to be exported in pixels.
imageY	Physical height of the image to be exported in pixels.
imageTransparency	Signature image background transparency as Boolean.
imageScale	Whether to scale the image or not as Boolean.
maxUpScalePercent	Maximum allowed upscale percentage as float (0 to 100).
rawDataFormat	By default the SDK sends back the signature data as compressed SigString. In addition, raw data can be exported after signature capture as string. Use ENC as a value for this field for exporting signature raw data in eSign Emcee format as an encrypted base 64 string.
minSigPoints	Minimum number of points required to qualify the signer squiggle as a valid signature as a number.

Once the required JSON input message is formatted, you should pass it to the Chrome or Firefox extensions. It should be set as an attribute to the custom html event **SignStartEvent** raised by a Web page.

The following interface appears for signature capture. The toolbar will have options to Clear the Signature, Accept the Signature, and Cancel the signature. If connection with the signature pad device is successful, you will see the “Pad initialized, start signing” message in the status bar (at the bottom) of the window.

8.1.2 – OUTPUT Message

The SigPlusExtLite SDK sends back an output message in the following scenarios after the signature capture is initiated using an input command.

1. Failed to initialize the signature capture dialog due to incorrect input data or any other device/driver related failures.
2. User cancelled signing.
3. User accepted a signature.
4. Signature not captured/accepted.

The output message has a ‘status’ Boolean parameter indicating whether the signature capture is successful or not. Chrome applications can rely on this parameter to identify if a signature is captured or not.

Here is a sample output message

{"isSigned": true , "imgData": "Base64 formatted image data" , "rawSigData": "Base64 formatted raw signature data inn Encrypted eSign Emcee format", "sigstring": "Signature data in compressed SigString format ", "padInfo": "Connected signature pad information"};

Parameter	Description
isSigned	Parameter value specifies whether a signature is captured or not as Boolean. Returns true if user accepted a signature and signature capture is successful. A positive result has succeeding image and raw signature data payloads in the message, otherwise they will be empty strings.
imgData	Signature image as base 64 string data. The format will be either JPG or PNG as specified in the input message. Contains the empty string if the signature is not captured.
rawSigData	Raw Signature data as eSign Emcee Encrypted Base 64 String. Contains an empty string if the signature is not captured.
sigstring	Compressed signature data in Topaz SigString format.
padInfo	Contains the information about the signature pad used for signature capture.
errorMsg	In case of any failure or user cancelled signing, this field contains the error message. Client applications can use this info to resolve the errors.

The OUTPUT message is sent back as an event attribute for the event ***SignResponse*** raised by the extensions and handled by the web page.

8.2 – Signature Capture and Custom Signature Image Generation

The Topaz SigPlusExtLite SDK supports generation of drawing custom text beside the signature squiggle. The SDK has APIs to specify the custom text to be drawn and percentage of image width to be used for drawing text.

8.2.1 – INPUT Message

Here is a sample JSON string containing all the supported INPUT for exporting custom images:

```
{"firstName": "John" , "last Name": "Doe" , "eMail": "johndoe@demo.com" , "location": "LosAngels,CA" , "imageFormat": 2 , "imageX": 300 , "imageY": 150 , "imageTransparency": true , "imageScaling": false, "maxUpScalePercent": 0.0, "rawDataFormat": "ENC" ,  "minSigPoints":25, "displayCustomTextDetails":true, "customTextPercent":50, "customTextLine1":"This is for demo purpose only", "customTextLine2": "This is for demo purpose only" };
```

Parameter	Description
firstName	Signature Detail First Name as String.
lastName	Signature Detail Last Name as String.
eMail	Signature Detail Email as String.
location	The signing location as String, could be a physical location or longitude and latitude.
imageFormat	Format of the signature image to be exported after signature capture. Send 1 for JPG and 2 for PNG. The default format is PNG.
imageX	Physical width of the signature image to be exported in pixels.
imageY	Physical height of the image to be exported in pixels.
imageTransparency	Signature image background transparency as Boolean.
imageScale	Whether to scale the image or not as Boolean.
maxUpScalePercent	Maximum allowed upscale percentage as float (0 to 100).
rawDataFormat	By default the SDK sends back the signature data as compressed SigString. In addition, raw data can be exported after signature capture as string. Use ENC as a value for this field for exporting signature raw data in eSign Emcee format as an encrypted base 64 string.
minSigPoints	Minimum number of points required to qualify the signer squiggle as a valid signature as a number.
displayCustomTextDetails	Display custom text beside signature squiggle as Boolean.
customTextPercent	Width of the signature image to be used for drawing custom text as percentage. Min and Max percentage values are 20 and 50. Any value specified below 20 will be defaulted to 20 and any value specified above 50 is defaulted to 50.
customTextLine1	Custom text line one as a String
customTextLine2	Custom text line two as a String.

Once the required JSON input message is formatted, you should pass it to the Chrome or Firefox extensions. It should be set as an attribute to the custom html event **SignStartEvent** raised by a Web page.

The following interface appears for signature capture. The toolbar will have options to “Clear the signature”, “Accept the signature”, and “Cancel the signature”. If the connection with the

signature pad device is successful, you will see the “Pad initialized, start signing” message in the status bar (at the bottom) of the window.

8.2.2 – OUTPUT Message

The SigPlusExtLite SDK sends back an output message in the following scenarios after the signature capture is initiated using an input command.

1. Failed to initialize the signature capture dialog due to incorrect input data or any other device/driver related failures.

2. User cancelled signing.

3. User accepted a signature.

4. Signature not captured/accepted.

The output message has a ‘status’ Boolean parameter indicting whether the signature capture is successful or not. Chrome applications can rely on this parameter to identify if a signature is captured or not.

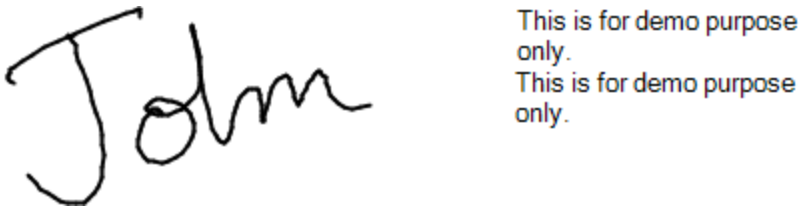
Here is a sample output message

{"isSigned": true , "imgData": "Base64 formatted image data" , "rawSigData": "Base64 formatted raw signature data inn Encrypted eSign Emcee format", "sigstring": "Signature data in compressed SigString format ", "padInfo": "Connected signature pad information"};

Parameter	Description
isSigned	Parameter value specifies whether a signature is captured or not as Boolean. Returns true if user accepted a signature and signature capture is successful. A positive result has succeeding image and raw signature data payloads in the message, otherwise they will be empty strings.
imgData	Signature image as base 64 string data. The format will be either JPG or PNG as specified in the input message. Contains the empty string if the signature is not captured.
rawSigData	Raw Signature data as eSign Emcee Encrypted Base 64 String. Contains an empty string if the signature is not captured.
sigstring	Compressed signature data in Topaz SigString format.
padInfo	Contains the information about the signature pad used for signature capture.
errorMsg	In case of any failure or user cancelled signing, this field contains the error message. Client applications can use this information to resolve the errors.

The OUTPUT message is sent back as an event attribute for the event ***SignResponse*** raised by the extensions and handled by the web page.

Here is a sample image with custom text:



9.0 – End User Deployment

Once the Topaz SigPlusExtLite SDK integration is completed, the next step is to deploy the required software on end user machines. Follow the steps outlined in the SigPlusExtLite User Installation Guide at: **www.topazsystems.com/Software/SigPlusExtLite_UserInstall.pdf**.