SigCaptureWeb SDK Guide

Version 1.2
May 15, 2020

Table of Contents

1.0 – Introduction

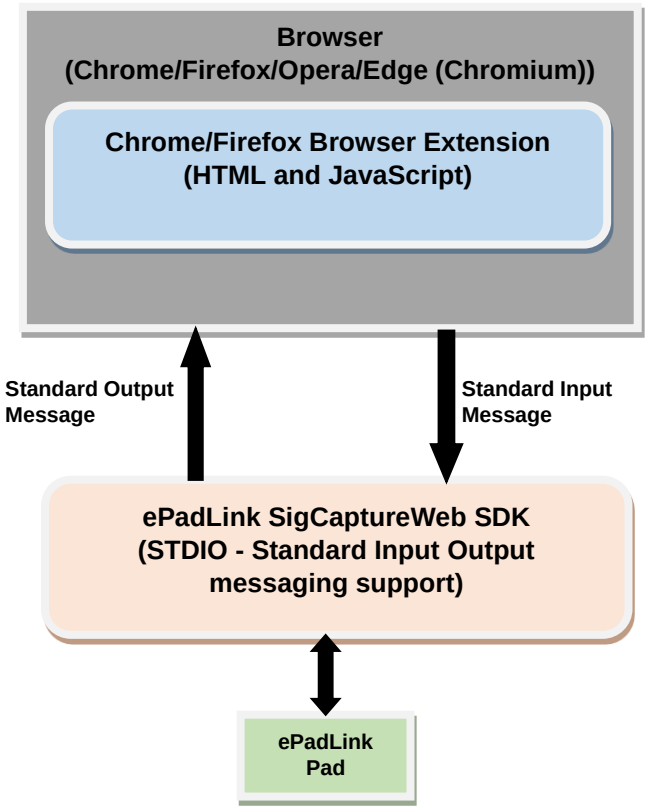
The ePadLink SigCaptureWeb SDK offers a mechanism and platform for developers and integrators to capture handwritten signatures securely for web applications running in the Google Chrome, Mozilla Firefox, Opera, and Microsoft Edge (Chromium) browsers. The SDK provides capabilities for capturing secured biometric handwritten signatures using electronic signature pads from ePadLink.

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. The biometric signature data for the captured signature is obtained in encrypted form and can be saved in any database for future use.

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 can be used for signature capture. Browser extensions are the best option for enabling web pages with the capability to capture signatures using the devices connected to client desktops.

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



2.1 – ePadLink SigCaptureWeb SDK

The ePadLink SigCaptureWeb SDK has been developed as a standard C#.Net application. It has a built–in mechanism to capture signatures using ePadLink signature capture devices and also to expose the raw biometric signature data in ENC or SIG format and the signature as an image (JPG, PNG, etc.). It has all the input and output interfaces implemented as Standard Input and Output streams as required by the Chrome and Firefox Browser Extension frameworks. The SDK processes the input text messages from the 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, launches it through Google Connect APIs or Web Extension APIs, and sends a notification back to the Extension when the application is ended by the user.

2.2 – Chrome and Firefox Extension/Webpage

Chrome and Firefox Extensions are HTML, JavaScript, and CSS based code modules that are launched during startup of the browser or launched on demand from web page JavaScript. Both extensions use JavaScript based Chrome Native Messaging APIs to launch and communicate with the ePadLink SigCaptureWeb SDK for signature capture and other relevant features. The extension listens for the output messages from SigCaptureWeb and processes them accordingly. Chrome Native Messaging has a Connect API to launch the applications (which can process the standard input and output messages) 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, Chrome Native Messaging APIs have mechanisms to send input messages to the SigCaptureWeb SDK and receive output messages from applications.

The Opera and Edge browsers also support installation and use of Chrome extensions. ePadLink SigCaptureWeb SDK leverages this to capture signatures within Opera and Edge browsers.

3.0 – Key Features

The ePadLink SigCaptureWeb SDK provides the following features:

- Set the minimum number of signature points required to qualify a signature as valid
- Set the pen thickness and color
- Initiate signature capture
- Clear the captured signature
- Get information about the connected Signature Pad
- Export a signature image PNG/JPG (base64 string)
- Export the raw biometric signature data in eSign Emcee or. Sig formatted encrypted base 64 string.

4.0 – Operating Systems Supported

The ePadLink SigCaptureWeb SDK can be integrated into web pages running in the latest versions of Google Chrome, Mozilla Firefox, Opera, and Microsoft Edge (Chromium) browsers installed on Windows 7/8.1/10 32-bit operating systems. For 64-bit Windows operating systems running the 64-bit Chrome and Firefox browsers, the SDK should be run as a 32-bit application.

The samples have been tested in the latest version of Chrome, Firefox, Opera, and Edge (Chromium) browsers. Hence, it is recommended that you install the latest version of the Google Chrome, Mozilla Firefox, Opera, and Microsoft Edge (Chromium) browsers.

***Note**: NET Framework 3.5 or higher should be available on the end users Windows computer.*

5.0 – Signature Capture Devices

The ePadLink SigCaptureWeb SDK supports capturing signatures using electronic signature pads from ePadLink. Universal Installer based drivers have to be installed for the ePadLink SigCaptureWeb SDK to capture signatures using signature pad devices.

6.0 – Installation Steps

Installation of ePadLink SigCaptureWeb requires two steps. First, the ePadLink SigCaptureWeb SDK and the relevant drivers must be installed. Then the ePadLink SigCaptureWeb Browser Extensions must be installed.

6.1 – Install ePadLink SigCaptureWeb SDK

Before the extensions are installed, the ePadLink SigCaptureWeb SDK must be installed on the client machines. To install the ePadLink SigCaptureWeb SDK on your computer follow the following steps:

1. Download and save the ePadLink SigCaptureWeb installer to your computer's desktop from the following URL: www.epadlink.com/SigCaptureWeb/SigCaptureWeb.exe
2. Right-click on the "SigCaptureWeb.exe" installer and choose "Run as Administrator". Then, follow the steps in the setup wizard.
3. Install the "Universal Installer" on your PC to provide the communication between your application programs and your ePadLink electronic signature pad. You can download and run the "Universal Installer" from the following URL: www.epadlink.com/universal-installer.html.

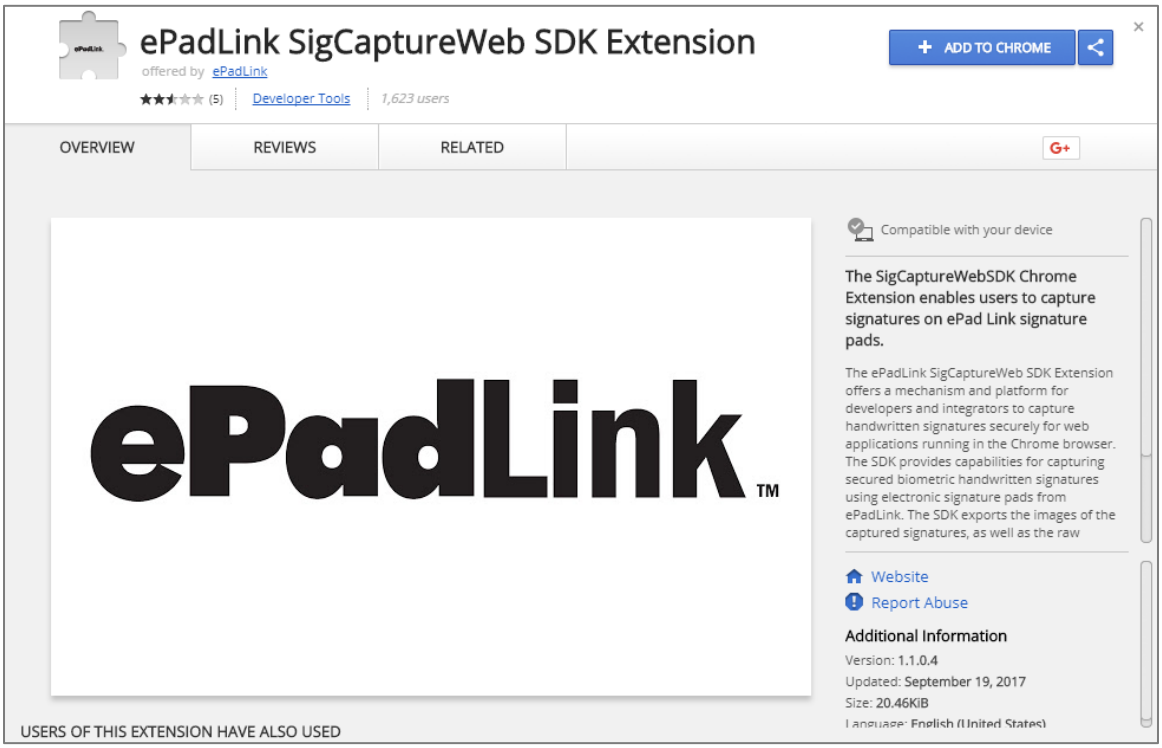
6.2– ePadLink SigCaptureWeb Browser Extensions

6.2.1 – Google Chrome

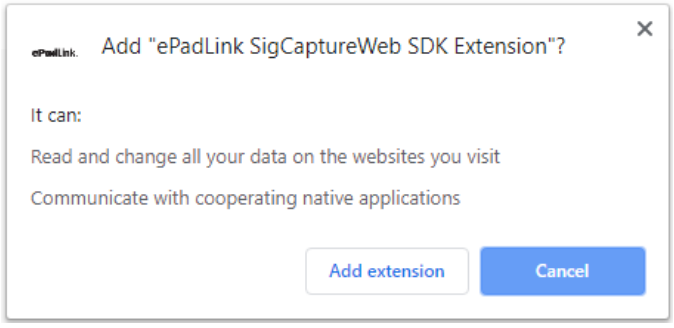
1. Start the Chrome Browser
2. Go to the "ePadLink SigCaptureWeb SDK Extension" page: https://chrome.google.com/webstore/detail/epadlink-sigcaptureweb-sd/idldbjenlmipmpigmfamdlfifkkeaplc

   ***Note**: In case clicking the URL does not work, try copy pasting the URL in the Chrome browser.*

3. Click the “ADD TO CHROME” button displayed on the top right of the page that is displayed (see the screenshot below).



4. Click on the “Add extension” button in the confirmation dialog that is displayed.



6.2.2 – Mozilla Firefox

To install the ePadLink SigCaptureWeb SDK extensions in the Firefox browser, please contact ePadLink support at support@epadlink.com.

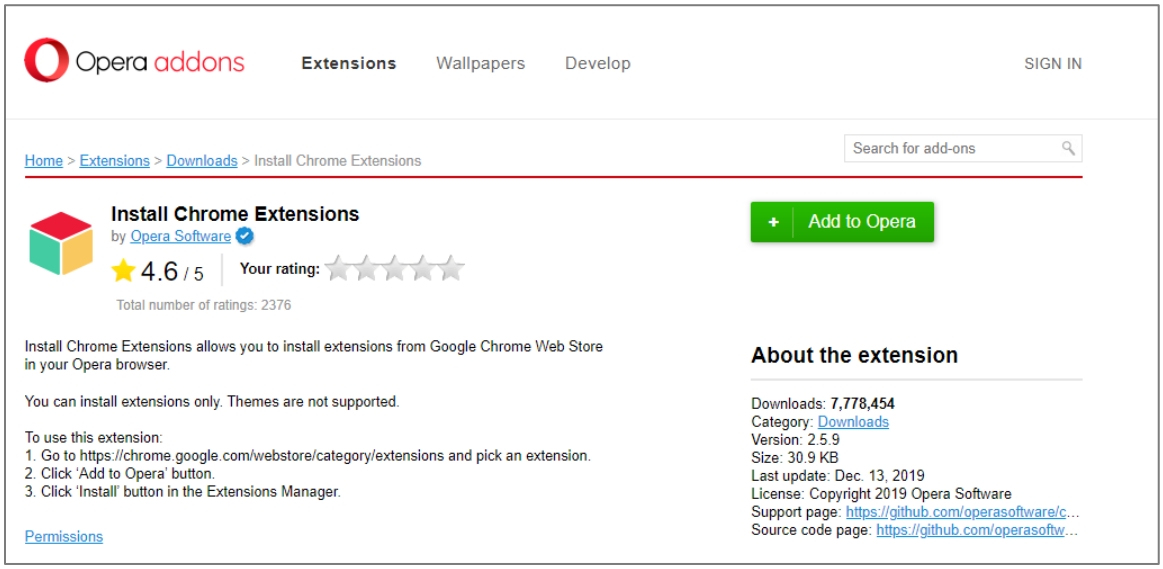
6.2.3 – Opera

Opera supports the installation of Chrome Extensions from the Chrome store. ePadLink SigCaptureWeb leverages the Chrome Extension to capture signatures in the Opera web browser. Follow the following steps to install the Chrome extension in the Opera web browser.

1. Start the Opera Browser.

2. Go to the Opera Add-ons page and install the “Install Chrome Extensions” Add-on by navigating to the following URL: https://addons.opera.com/en/extensions/details/download-chrome-extension-9/?display=en.

3. Click the “+ Add to Opera” button displayed on the top right of the page (see the screenshot below). To view the installed extension, type Ctrl+Shift+E.

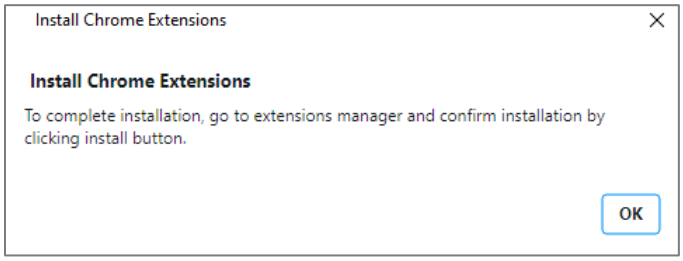


4. Go to the ePadLink SigCaptureWeb SDK Extension page on the Chrome web store: https://chrome.google.com/webstore/detail/epadlink-sigcaptureweb-sd/idldbjenlmipmpigmfamdlfifkkeaplc

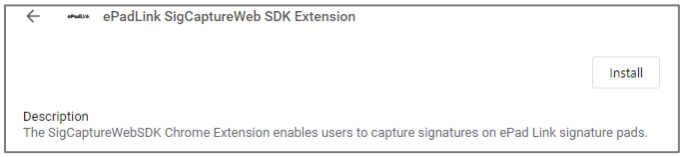
   ***Note**: In case clicking the URL does not work, try copy pasting the URL in the Opera browser.*

5. Click on the “Add to Opera” button displayed on the top right of the page that is displayed (see the screenshot below).

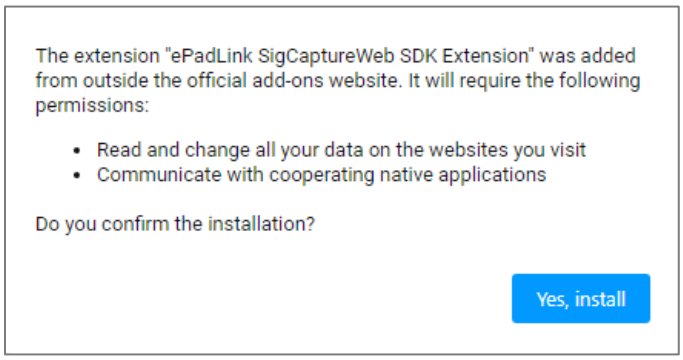
6. Click OK in the “Install Chrome Extensions” dialog that is displayed.



7. Click on the “Install” button on the top right of the displayed page to install and enable the extension (see the screenshot below).



8. Finally, click on the “Yes, install” button in the confirmation dialog that is displayed (see the screenshot below).



6.2.4 – Microsoft Edge (Chromium)

Microsoft Edge has adopted the Chromium open source project. It now supports installation of Chrome Extensions, and ePadLink SigCaptureWeb leverages the Chrome Extension to capture signatures in the Microsoft Edge browser. Follow the following steps to install the Chrome extension in the Microsoft Edge browser.

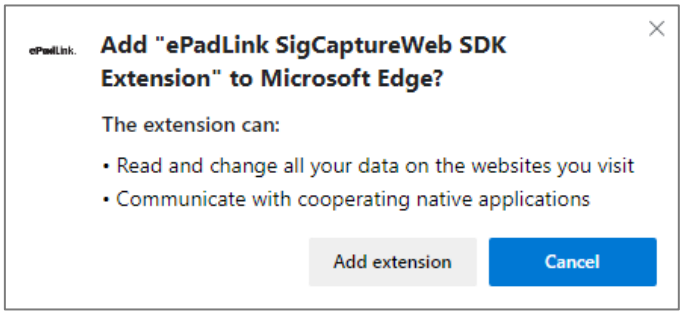
1. Start the Microsoft Edge (Chromium) Browser.

2. Go to the "ePadLink SigCaptureWeb SDK Extension" page by navigating to the URL: https://chrome.google.com/webstore/detail/epadlink-sigcaptureweb-sd/idldbjenlmipmpigmfamdlfifkkeaplc

   ***Note**: In case clicking the URL does not work, try copy pasting the URL in the Edge browser.*

3. A window showing the Chrome Extension similar to the one in Section 6.2.1 will be displayed. Click the "Add to Chrome" button displayed on the top right of the page (see the screenshot below).

4. Click on the "Add extension" button in the confirmation dialog that is displayed.



6.3 – Run the Sample Web Page

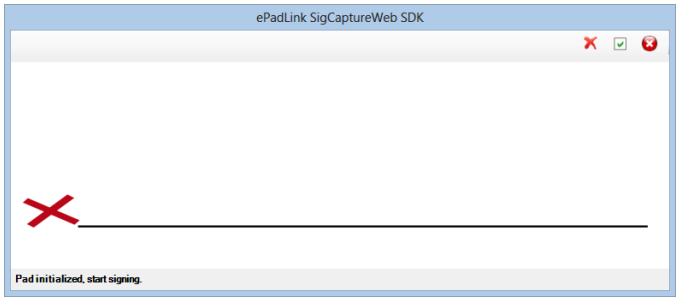
After making sure the ePadLink SigCaptureWeb SDK software is installed (see Section 6.1) and the signature pad is connected to the PC, open this page in your preferred browser: www.epadlink.com/sign_chrome_ff_sigcapturewebsdk/sign_chrome_ff_sigcapturewebsdk.html

Click on the "Sign" button to display the signature capture window. The signature window user interface is explained in detail in the next section.

7.0 – Signature Capture Window Interface

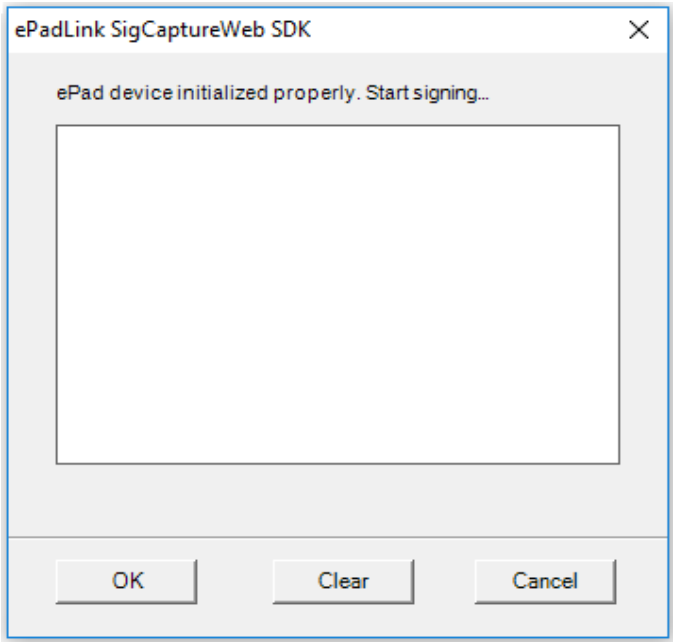
The signature capture window interface can be controlled using the configuration parameter “uitype” in the configuration XML. Currently two different types of capture windows are supported – Modern and Classic.

The “Modern” UI looks like the following:



Sign on your signature pad, and the signature will appear in the signature capture window. To re-sign, click the red “X” in the upper right of the window. To accept the signature, click the green check mark icon. To cancel, click the red circle with the “X” in it.

The “Classic” UI looks like the following:



Sign on your signature pad, and the signature will appear in the signature capture window. To re-sign, click the “Clear” button. To accept, click the “OK” button. To cancel, click the “Cancel” button.

To control the signature capture window interface, follow the steps given below.

1. Go to the location where the SDK is installed:

   - For Chrome, Opera and Microsoft Edge (Chromium), typically the path will be “C:\Program Files (x86)\SigCaptureWeb SDK\Chrome”
   - For Firefox, typically the path will be “C:\Program Files (x86)\SigCaptureWeb SDK\Firefox”

2. Find the “ePadLink.SigCaptureWebSDK.exe” XML configuration file in the SDK folder.

3. Open the configuration file in a text editor to change the configuration parameter. The contents of the configuration file will be as follows:

```
<? xml version = "1.0" encoding = "utf-8" ?>
    <configuration>
        <appSettings>
            <add key="uitype" value="1" />
        </appSettings>
    </configuration>
```

4. Change the value of the key “uitype” in the configuration file.

```
<add key="uitype" value="1" />
```

5. To display the “Modern” signature capture window the value must be set to “1”. For the “Classic” signature capture window the value must be set to “2”. By default, the “Modern” signing window will be displayed.

8.0 – SigCaptureWeb SDK Integration for Signature Capture in Web Browsers

For web pages running in the supported browsers, the only required step is to raise and listen for predefined custom HTML events within the web page.

8.1 – Launching the Extensions from a Webpage

The SigCaptureWeb SDK extensions rely on custom HTML events for communication between the web page and the extensions and vice versa.

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

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

The following code snippet demonstrates raising the custom HTML event "SigCaptureWeb_SignStartEvent" to initiate signature capture and also register and implement the "SigCaptureWeb_SignResponse" event for processing the output from the SDK. Input and output messages are passed as event attributes. The input to the SDK must be set using the attribute name "SigCaptureWeb_MsgAttribute" and response from the SDK must be read from event attribute "SigCaptureWeb_msgAttri"

```
var message = { "firstName": "", "lastName": "", "eMail": "", "location": "",
"imageFormat": 1, "imageX": imgWidth, "imageY": imgHeight, "imageTransparency": false,
"imageScaling": false, "maxUpScalePercent": 0.0, "rawDataFormat": "ENC", "minSigPoints":
25, "penThickness": 3, "penColor": "#000000" };
document.addEventListener('SigCaptureWeb_SignResponse', SignResponse, false);
var messageData = JSON.stringify(message);
var element = document.createElement("SigCaptureWeb_ExtnDataElem");
element.setAttribute("SigCaptureWeb_MsgAttribute", messageData);
document.documentElement.appendChild(element);
var evt = document.createEvent("Events");
evt.initEvent("SigCaptureWeb_SignStartEvent", true, false);
element.dispatchEvent(evt);
function SignResponse(event) {
	var str = event.target.getAttribute("SigCaptureWeb_msgAttri");
	var obj = JSON.parse(str);
	SetValues(obj, imgWidth, imgHeight);
}
```

9.0 – Signature Capture and Data Export

As the SigCaptureWeb 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.

9.1 – Signature Capture

The ePadLink SigCaptureWeb SDK supports only one input message which will trigger the signature capture using the ePadLink 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 or SIG), width and height of the image, background image transparency, etc.) for export after the capture and other parameters described in the next section.

9.1.1 – INPUT Message

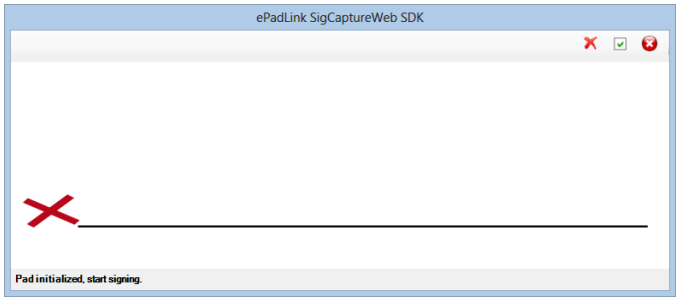
Here is a sample JSON string containing all the supported INPUT.

```
{ "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, "penThickness": 3, "penColor": "#000000" };
```

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.
imageScaling	Whether to scale the image or not as Boolean.
maxUpScalePercent	Maximum allowed upscale percentage as float (0 to 100).
rawDataFormat	Raw data can be exported after signature capture as string. Use ENC as value for this field for exporting signature raw data in eSign Emcee format and SIG to export it in Topaz .Sig format.
minSigPoints	Minimum number of points required to qualify the signer squiggle as a valid signature as a number.
penThickness	Thickness of the pen to be used for drawing the signature on the screen and image to be exported. Though the API accepts higher values, the recommended values are in the range of 1 to 7 depending on the resolution of the image to be exported.
penColor	Color of the pen to be used for drawing the signature on the screen and image to be exported. The value should be passed as a Hexadecimal RGB value. The RGB value must be prefixed with '#'.

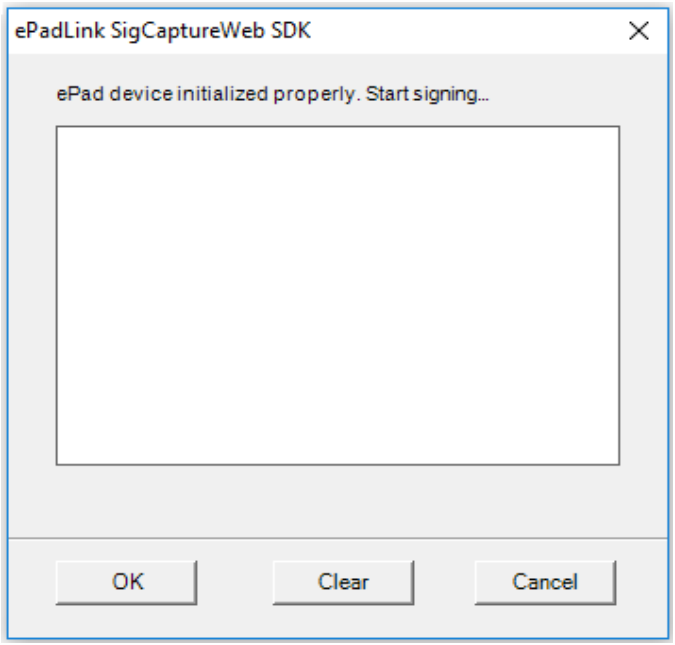
Once the required JSON input message is formatted, you should pass it to the Chrome or Firefox extension. It should be set as an attribute to the custom html event **SigCaptureWeb_SignStartEvent** raised by a Web page.

The following interface appears for signature capture by default. 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.



The toolbar will have options to Clear the Signature, Accept the Signature, and Cancel the signature.

If the configuration parameter “uitype” value is set to “2” (see Section 7.0), the following “Classic” interface will appear for signature capture. If connection with the signature pad device is successful, you will see the “ePad device initialized properly. Start signing…” message on the signature capture window.



The signature capture window will have options to Clear the Signature, Accept the Signature (“OK”), and Cancel the signature.

9.1.2 – OUTPUT Message

The SigCaptureWeb 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. Applications can rely on this parameter to identify if a signature is captured or not.

The following is a sample output message:

```
{ "isSigned": true, "imgData": "Base64 formatted image data", "rawSigData": "Base64 formatted raw signature data in Encrypted eSign Emcee or .Sig 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 or as. Sig formatted string. Contains an empty string if the signature is not captured.
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 **SigCaptureWeb_SignResponse** raised by the extensions and handled by the web page.

10.0 – End User Deployment

Once the ePadLink SigCaptureWeb SDK integration is completed the next step is to deploy the required software on end user machines. Follow the steps in Section 6.0.