CDS Global Browser Encryption

--- a plugin for payment processing with RSA Encryption

Version 1.0.0

08/16/2017

CDS Global Browser Ecnryption API Integration

REVISION HISTORY

Version	**Action**	**Description of Change**	**Date**
1.0.0	New	Base version	08/16/2017

Table of Contents

Introduction: -

Purpose

The Browser Encryption integration document provides the functional and technical details of browser Encryption offered by CDS Global. This document will serve as the guide during the development of integration with CDS Global Browser Encryption plugin.

Scope

The scope of this integration is to use CDS Global Browser Encryption to make the payment secure. At a high-level CDS Global Browser Encryption will encrypt the credit card number entered by the user on payment page such that whoever integrates this would no longer need to worry about transmitting raw credit card number over the network instead they can use the cipher generated by browser Encryption. This plug in will act on credit card data entered by the user in following way:

- Validation
- Any valid credit card number is encrypted as soon as user enters valid credit card data
- Secure and high availability

Definitions, Acronyms and Abbreviations

- **HTTP** – Hypertext Transfer Protocol
- **WWW** – World Wide Web
- **AJAX -** Asynchronous JavaScript and XML

Integration Protocols & Message

Integration Protocol: HTTP over SSL

Integration request: Method = GET

Response: - bytes

Integration Method: Include the <script> tag with corresponding source in html

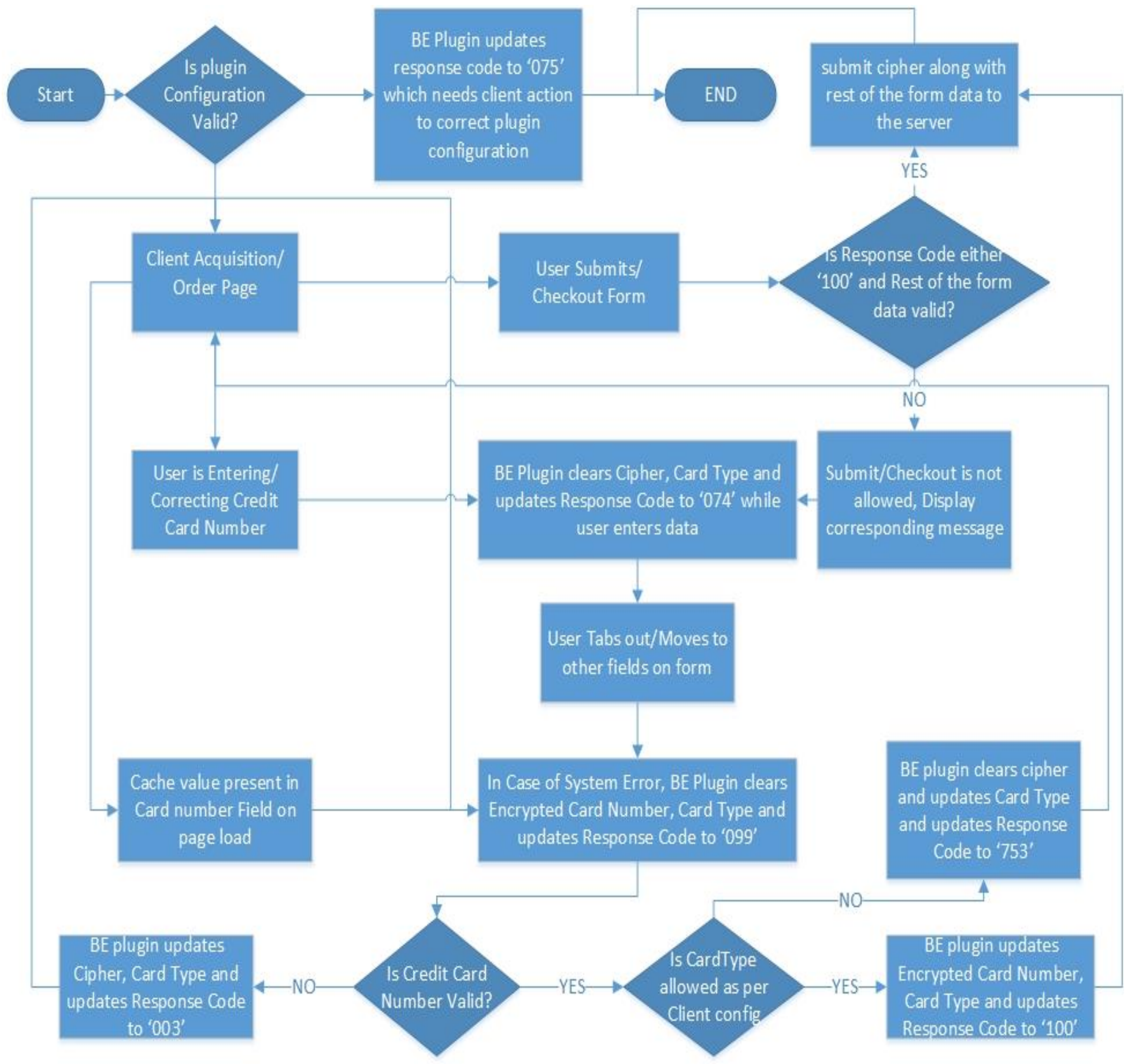
Production Server

BA Test Server

The script file will be available in AWS over HTTPS for public access to integrate CDS Process for secure payment processing using RSA encryption.

Client can access the script file using get request and can include plugin in corresponding html page for encryption.

Browser Encryption FLOW: -


Start
Is plugin Configuration Valid?
BE Plugin updates response code to '075' which needs client action to correct plugin configuration
END
submit cipher along with rest of the form data to the server
YES
Client Acquisition/ Order Page
User Submits/ Checkout Form
Is Response Code either '100' and Rest of the form data valid?
NO
User is Entering/ Correcting Credit Card Number
BE Plugin clears Cipher, Card Type and updates Response Code to '074' while user enters data
Submit/Checkout is not allowed, Display corresponding message
User Tabs out/Moves to other fields on form
Cache value present in Card number Field on page load
In Case of System Error, BE Plugin clears Encrypted Card Number, Card Type and updates Response Code to '099'
BE plugin clears cipher and updates Card Type and updates Response Code to '753'
NO
BE plugin updates Cipher, Card Type and updates Response Code to '003'
NO
Is Credit Card Number Valid?
YES
Is CardType allowed as per Client config
YES
BE plugin updates Encrypted Card Number, Card Type and updates Response Code to '100'

Note: - '099' and '074' response codes are generally not reported expect in case of unusual failure of BE Plugin and '099' could be also be due to client code not being registered with CDS so it is still recommended to include these codes to isolate/identify any error with plugin.

Integration Recommendations and Steps: -

Recommendations

HTTPS Recommendation**: -**

All payment data submitted should be made via a secure HTTPS connection to avoid any man-in-middle attacks.

Integration steps: -

- **Step 1**: - Including Browser Encryption plug in and configuration

Below Script Tags should be included in the same order as below either along with bundling of your script files or include these in script tag in your payment processing page:

a. Client page should include a script tag with source pointing to the CDS Process script file.

Ex: `<script src="https://s3.amazonaws.com/cds-tzn-test/resources/cds-process.min.js"></script>`

b. Client page should include a script tag that has configuration to initialize couple of variables to the CDS process script after including CDS process script file as below

i. Below variable should be initialized with an array of supported Card Types by the client

```
CDS.cdsProcess.allowedCards = [ 'MC', 'VI', 'AX' ];
```

Master Card Type: 'MC'
Visa Card Type: 'VI'
Amex Card Type: 'AX'
JCB Card Type: 'JCB'
Diners Card Type: 'DC'
Discover Card Type: 'DI'

Note: - Only above card types are supported by this plugin

ii. Below variable should be initialized with client code agreed with CDS as follows

```
CDS.cdsProcess. clientCode("someClientCode");
```

Ex:

```
<script>
   CDS.cdsProcess.allowedCards = [ 'MC', 'VI', 'AX' , 'DI', 'DC', 'JCB'];
   CDS.cdsProcess.clientCode('abc');
</script>
```

Note: - It is recommended to have the above scripts be executed before handling any user action on the payment page.

- **Step 2**: - Payment form with Data attributes for Browser Encryption to access card details

Client page should have below HTML5 data attributes included for corresponding elements irrespective of the way they are presented to the user:

1. Credit Card number input field

   The field that takes credit card number from the user should be an input element and it should have a data attribute *(i.e data-cds = "ccNumber")* such that included script can have access to the credit card number entered by the user.

   Ex: `<input id="cc-number" type="text" class="cc-number" data-cds="ccNumber" required>`

- **Step 3**: - Payment form with Data attributes on Hidden Fields for Browser Encryption to render data

Optional: -

This optional step is needed if client page integrating with this plugin wants to read the cipher data, card Type, response code from hidden fields based on change events

Client page should have the following HTML5 data attributes specified for corresponding elements irrespective of the way they are presented on the page:

1. Cipher input field to render back

   A hidden input field must be specified in the client page with data attribute (i.e data-cds = *"cipher"*) such that encrypted card number can be rendered back by script

   Ex:

   `<input id="cipher" type="hidden" placeholder="Cipher" data-cds = "cipher" readonly>`

2. Card Type input field to render back

A hidden input field must be specified in the client page with data attribute (i.e data-cds = " *cardType*") such that type of card based on the number entered by the user in credit card number field can be rendered back by script

Ex:

```
<input id="cardType" type="hidden" placeholder="Card Type" data-cds="cardType"
readonly>
```

3. Response code input field to render back

A hidden input field must be specified in the client page with data attribute (i.e data-cds = "*responseCode*") such that any change of validation/Encryption status due to any action on credit card number or Expiration date can be rendered back by script

Ex:

```
<input id=" responseCode" type="hidden" data-cds = " responseCode" readonly>
```

One of the valid response codes is rendered to hidden input field as user enters credit card number and proceeds to rest of the form/ submits form (refer appendix for valid response codes)

- **Step 4**: - Based on Step 3, Capture encrypted card number and response Code from Browser Encryption plugin

If Step 3 is done Listen to change events on hidden fields (i.e. 4.i) or else it is also just enough for you to listen to custom events on "CDS.cdsProcess.cdsCard" (i.e. 4.ii) without need of having hidden fields on your page

4.i) Listen to change events on hidden fields: -

As Browser Encryption does the encryption process it keeps the data rendered to the hidden fields specified on the client page up to date for the client to capture data.

Client can listen to change events through change event listeners on corresponding hidden fields to capture either response code, card type, the encrypted card number generated for the card number entered by the user or any kind of errors occurred during browser encryption process.

It is recommended to listen to response code field changes as it is a recommended way to determine the browser encryption process on the card details entered by the user before reading any other field data.

It is also recommended to check whether the response code hidden field has response code of ‘075’ for the first-time page is loaded so that you can capture any configuration errors. This is needed as change event listeners on hidden fields may not be registered by the time of change event getting triggered for reporting client configuration error.
Client can listen for response code change to “100” i.e Successful encryption for reading encrypted card number from the cipher hidden field which can be used for payment submission instead of credit card number.

Sample code for reading response code: -

<u>Change Event listener</u>: -

It is recommended to have the change event listener and check for response code ‘075’ inside “DOMContentloaded” event to ensure hidden fields are already rendered to the page: -

```
document
 .addEventListener(
       "DOMContentLoaded",
        function () {

            if (document.querySelector('[data-cds =
               responseCode"]').value == "075") {
                   handleValidationDisplay();
                   //Appropriate client desired handler can be
                   defined
             }

            document
               .querySelector('input[data-cds = "responseCode"]')
               .addEventListener('change', function() {
                     handleValidationDisplay();
                      //Appropriate client desired handler can be
                      defined
            });
}
);
```

Handler Function: - (i.e `handleValidationDisplay in the above sampleCode`)

As any change to response code would call this handler, client can make use of response code to show some validation error message such as ‘invalid credit card number’, ‘system error’, ‘please enter credit card number’ etc.

Sample Code: - Below function is only a sample one it is up to client to handle the response code in their own way

```
handleValidationDisplay = function() {

var responseCode =
       document.querySelector('[data-cds =
"responseCode"]').trim();
       if (responseCode == '100') {
       //Successful encryption, client can read the
       // data rendered by plug in
       // client can read the cipher from the hidden field
       // card type can be read from the card type hidden field
       } else if (responseCode == '003') {
       //Inform the user about invalid card number entered
       } else if (responseCode =='') {
       // Inform the user about card number field being empty
       } else if (responseCode == '753') {
       // Inform the user about card type is not allowed
       } else if (responseCode == '074') {
       // This is optional can be used if you want to have any
       functionality while the user is entering the credit card
       number
       } else if (responseCode == '075') {
       // Please correct the configuration provided for the
       plugin
       } else if (responseCode == '099') {
       // Please contact CDS to report this system error, this
       can also be due to client not being registered with CDS
       }
}
```

4.ii) Listen for special change events on “CDS.cdsProcess.cdsCard”

If you have done step 3 along with 4.i, this is optional as you are already reading values from hidden fields but in case you choose to not have hidden fields on your page you can do 4.ii along step 3 which does not affect your process.

Client can have a JavaScript listener on object (i.e “CDS.cdsProcess.cdsCard”)

It is recommended to have the “cdsCardValChange” listener event even before “DomContentLoaded” event triggered such that configuration error of response code ‘075’ is also captured at initial page load

Custom event name to which client must listen for any plugin response changes: “cdsCardValChange”

JavaScript Listener: -

```
CDS.cdsProcess.cdsCard.addEventListener(“cdsCardValChange”,
      function(cdsResponse) {
            if (document.readyState === "complete"
                  || (document.readyState !== "loading"
                  && !document.documentElement.doScroll)) {
               handleValidationDisplay(cdsResponse);

      //Appropriate client desired handler can be defined
            } else {
               document.addEventListener("DOMContentLoaded",
                  function() {
                  handleValidationDisplay(cdsResponse);
            //Appropriate client desired handler can be defined
                  });
            }
      });
```

Handler Function: - (i.e `handleValidationDisplay in the above sampleCode`)

As any change to response code would call this handler, client can make use of response code to show some validation error message such as ‘invalid credit card number’, ‘system error’, ‘please enter credit card number’ etc.

Sample Code: - Below function is only a sample one it is up to client to handle the response code in their own way

```
      handleValidationDisplay = function(cdsResponse) {

            var responseCode = cdsResponse.respCode.trim();
            if (responseCode == '100') {
               var cipher = cdsResponse.cipher;
               //cipher can be used instead of card number
               var cardType = cdsResponse.cardType;
               //card type can be used for validation with user
               entered card type
               } else if (responseCode == '003') {
                //Inform the user about invalid card number entered
```

```
                        } else if (responseCode == '') {
                        // Inform the user about card number field being empty
                       } else if (responseCode == '753') {
                        // Inform the user about card type is not allowed
                       } else if (responseCode == '074') {
                     // you can add any functionality you wish to have while
                     the user is entering the credit card number
                     } else if (responseCode == '075') {
                     // Please correct the configuration provided for the
                     plugin
                     } else if (responseCode == '099') {
                     // Please contact CDS to report this system error, this
                     can also be due to client not being registered with CDS
                     }
                }
```

- **Step 5**: - Based on step 3 and step 4, Submit the encrypted card number along with rest of the form data

  The last step is to add an event listener on form submission such that whenever user clicks on submit, if the response code is “100” (i.e successfully encrypted) you can submit form for payment processing with cipher along with rest of your form and prevent form submission in case of any other response code. (i.e ‘075’ or ‘099’ or ‘003’ or ‘753’ or ‘074’)

  Sample Code: -

```
form.addEventListener('submit',
    function(event) {
       event.preventDefault();
       captureAndHandleCDSResponse();
});
```

If Step 3 is done, to capture/collect validation && encryption response from the plugin user can just read the response from the hidden fields which will be loaded with values by the plugin as soon as user focuses out of the credit card number field below is sample code in section 5.i

5.i) Read plugin response from hidden fields

Sample Code: -

```
function captureAndHandleCDSResponse() {
       var cdsValCode = document
                  .querySelector('[data-cds = "responseCode"]').value;
       var cardTypeValue = $('input[data-cds="cardType"]');
       var cipherValue = $('input[data-cds = "cipher"]');
       if (cdsValCode == '100') {
                 cdsProcessResponseHandler(cdsValCode, cardTypeValue,
                                          cipherValue);
       } else if (cdsValCode == '074') {
             // This condition is just to retry for requesting response
```

```
                //which is mostly never reported at time of submission,
                //just a safety measure to also handle unexpected delay in
                //processing
                cdsValCode = CDS.cdsProcess.reportEncryptionResponse();
                if (cdsValCode == '100') {
                      cdsProcessResponseHandler(cdsValCode, cardTypeValue,
                                   cipherValue);
                } else {
                      handleValidationDisplay();
                }
          } else {
                handleValidationDisplay();
          }
    }

    function cdsProcessResponseHandler(cdsValCode, cardTypeValue,
                                   cipherValue) {
          // code to handle form submission to server along with cipher,
          //rest of the form data
    }
```

If client chooses to not have hidden fields it is also enough to call an API of the plugin in submit event listener which would trigger an ‘cdsCardRespChange’ event along with response inside plugin which can be handled as shown in sample code in section 5.ii

5.ii) Collect response from plugin through even listener: -

```
    function captureAndHandleCDSResponse() {
          CDS.cdsProcess.cdsCard
                .addEventListener('cdsCardRespChange',
                      function(cdsResponse) {
                         var cdsValCode = cdsResponse.respCode;
                         var cardType = cdsResponse.cardType;
                         var cipher = cdsResponse. cipher;
                            if (cdsValCode == '100') {
                                  cdsProcessResponseHandler(cdsValCode,
                                        cardTypeValue, cipherValue);
                            } else if (cdsValCode != '100') {
                               handleValidationDisplay(cdsResponse);
                            }
                            });
         CDS.cdsProcess.reportEncryptionResponse();
   }

    function cdsProcessResponseHandler(cdsValCode, cardTypeValue,
                                   cipherValue) {
          // code to handle form submission to server along with cipher,
          //rest of the form data
    }
```

Points to Note: -

1. ‘099’ and ‘074’ response codes are generally not reported expect in case of unusual failure of BE Plugin and ‘099’ could be also be due to client code not being registered with CDS so it is still recommended to include these codes to isolate/identify any error with plugin.

2. You have the flexibility to customize your logic based on the response code changes that can be observed on response code hidden field or change events on “CDS.cdsProcess.cdsCard”.

Browser Encryption Script Responsibilities: -

Captures and process the credit card number through client page as soon as user enters data and do corresponding processing based on user actions as follows: -

1. Validate the Card number as user enters credit card number field
2. Tracks and Provides the corresponding response code to the client page when user focuses out of the field or whenever requested
3. Determine the card type based on the data entered by the user
4. Track and Verify if the determined card type is supported by the client
5. Corresponding Validation code is provided
6. In case of successful validation encrypt the credit card number over client side for enhanced security
7. Provide the cipher along with success response code back to client
8. Provide card type based on the credit card number validation

The cipher provided can be used safely to send it to server instead of credit card number.

Integration summary: -

1. Include required script tags in corresponding html page having payment form
2. Add specified data attributes to the elements
3. Register event listeners for certain events to capture cipher either from hidden fields on payment page or through events fired
4. Handle captured response from plugin for using cipher instead of raw credit card number for sending it to the server

Restrictions with Browser Encryption: -

1. User is restricted to paste any data on to the credit card number field
2. User is not allowed to enter more than 19 digits
3. User can enter only numeric data in credit card number field

Summary of Browser Encryption process: -

- Validates the client configuration provided and any incorrect configuration is reported
- Continuously listens for user actions to enter credit card number
- Any changes to Card number will be continuously validated
- Whenever user enters valid credit card number it determines the type of card and validates whether the card type is one of allowed card types
- If validation is passed plugin encrypts the card number entered by the user
- Any changes to the credit card number will repeat the process

APPENDIX:

Browser Encryption Sample Response:

Sample Response rendered to the client page to corresponding hidden fields: -

Hidden Field	Description	Valid Values	Sample Response
Cipher	Encrypted card number for the credit card number entered		1492106301154446
Card Type	One of the supported card types	AX – AMEX DI – Discover DC – Diners Club JC - JCB MC – MasterCard VI –VISA	MC
Response Code	Response from Browser Encryption process	100 003 099 074 075 753 Blank	100

Browser Encryption Response Codes:

Response Code	Description	Action to take
100	Successful Encryption of credit card number	Read encrypted data from cipher hidden field then use it for payment processing
003	Invalid Card number entered	Do not allow form submission (optional) Prompt user to correct card number either immediately or when user choose to submit/checkout
Blank	Invalid or No data entered by user	Do not allow form submission (optional) Prompt user to not leave credit card number field empty
099	Encryption failure due to system error	Do not allow form submission Contact CDS for any system issues
074	User still entering credit card number	Request Browser Encryption process to report response and act accordingly
075	Either Client configuration is missing or Incorrect	Correct the client configuration with valid clientCode and valid allowed card types supported
753	Card number may be a valid number but it's card type is not allowed by the merchant	Do not allow form submission (optional) Prompt user to try some other card number as specific card type is not allowed.