Using the i4Go Technical Reference Guide

This guide provides an introduction to i4Go®, which supports PSD2 (Payments Service Directive 2) and SCA (Strong Customer Authentication), and reviews implementing i4Go using an Access Token and iFrame (i4m).

Introduction to i4Go

i4Go was designed to secure cardholder data (CHD) in e-commerce business environments with web browser-based applications that integrate online (website) and onsite (kiosk, Software as a Service) technologies by intercepting CHD at the point of entry—before it ever enters the merchant's Web server or hosting provider's system—and working with Shift4's PA-DSS validated Universal Transaction Gateway® (UTG®) or API Web service* and PCI DSS-compliant Lighthouse Transaction Manager payment gateway to replace the CHD with a TrueToken®, a unique ID to reference the actual data. This will drastically reduce e-commerce merchants' PCI DSS scope and may qualify them to use the SAQ-A (EP).

The application will use the TrueToken to process the transaction via UTG or API Web service and Lighthouse Transaction Manager. At no time does real CHD exist in the merchant's devices, applications, event logs, transport mechanisms, databases, Web servers, or hosting provider's systems.

*If the environment cannot support the use of a UTG, direct server-to-server Web service calls can be used to replace the CHD with a TrueToken and subsequently process the transaction.

WARNING! i4Go is designed to keep real and sensitive CHD information out of the merchant's Web server or hosting provider's system. If you send this information to the merchant's systems, you are defeating i4Go's purpose.

Security Best Practices

Please review these important details:

- Review the *Securing the Merchant's Site That Uses i4Go* document.
- Shift4 highly recommends that the merchant's server has an SSL certificate.

Other Implementation Requirements

Please review these important details, which are supplemental to the requirements outlined in the *Implementing i4Go* section:

- The application must be able to exchange an Auth Token for an Access Token. (For additional information, see the *Prior to Implementation* section.)
- The merchant's server must be able to perform a server-to-server call to obtain an access block. If the merchant's server communication package does not recognize Shift4's SSL certificate, a ZIP is available here under i4Go: https://myportal.shift4.com/index.cfm?action=support.security.
- For production, the merchant will need to supply the static, public IP address or range of IP addresses for each Web server to the Shift4 Installations team.
- Shift4 recommends language be added introducing i4Go to the merchant's end users and briefly explaining how it protects CHD. For your convenience, we have included content in each template which is hidden by default but can be displayed. We recommend you use the text that can be displayed in the templates. (For additional information, see *Appendix B*.)

Prior to Implementation

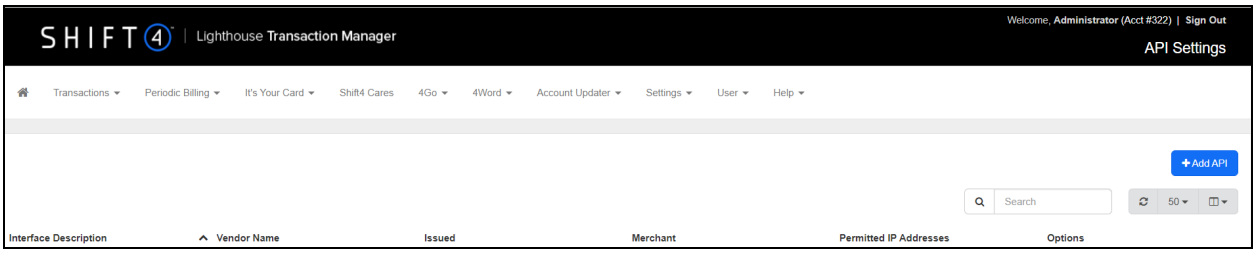
For developers that are implementing i4Go and Shift4 API integration, the Shift4 API team will provide the developer with the Auth Token during the certification process.

For certification and production, an Auth Token will need to be exchanged for an Access Token. (See RESTful API in MyPortal API Corner for instructions on Access Token Exchange.)

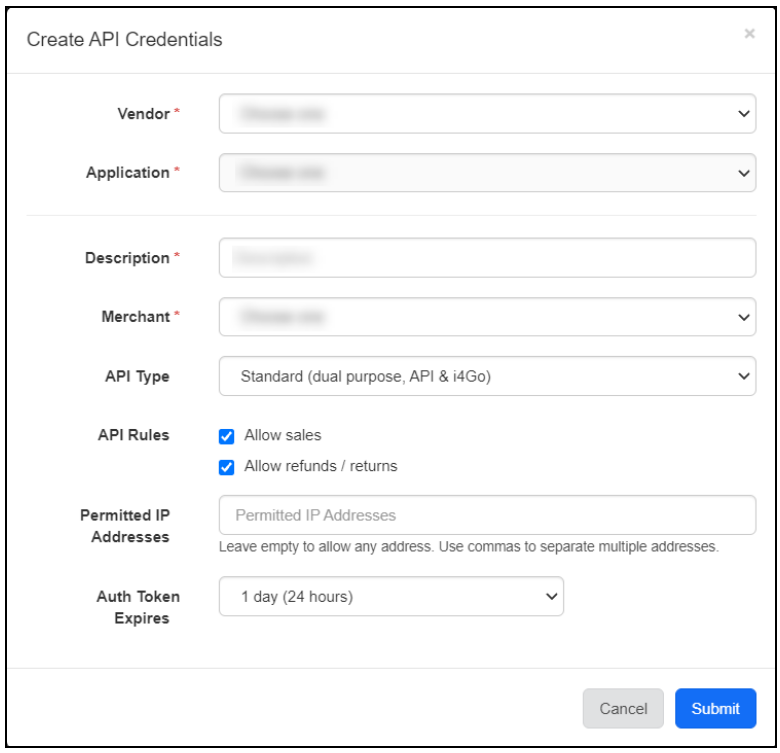
Note: If only i4Go is being implemented and you are not planning to use the Shift4 API, it may be possible to generate an Access Token instead of generating an Auth Token and exchanging it for an Access Token. For additional information, contact your Shift4 API support analyst.

The process for generating an Auth Token for production is briefly outlined below:

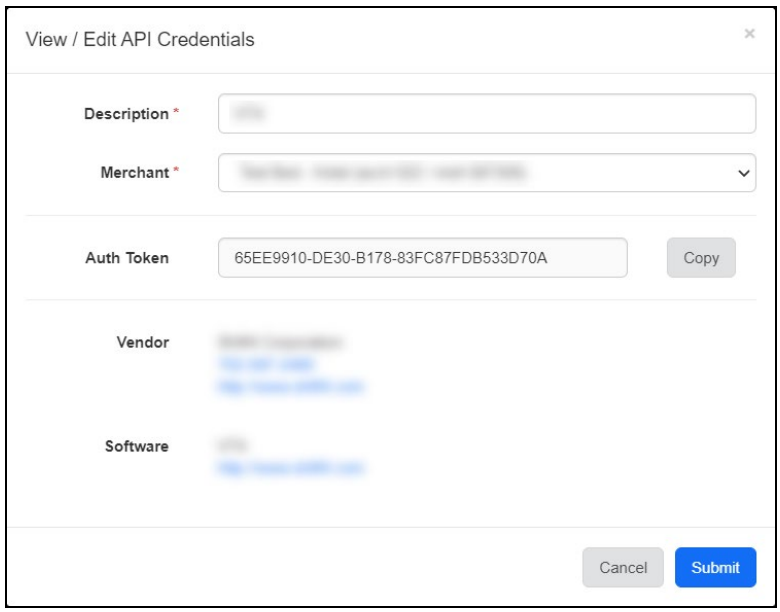
- The Account Administrator signs in to Lighthouse Transaction Manager.
- From the menu, the Account Administrator selects **Settings > API Settings**.
- On the API Settings page, the Account Administrator clicks **Add API** to manually create API credentials.



- In the **Create API Credentials** window, the Account Administrator configures the applicable options and clicks **Submit**.



- In the View/Edit API Credentials window, the Account Administrator records and provides the Auth Token to the application and clicks **Submit**.



Note: For additional information on this process, the Account Administrator can sign in to Lighthouse Transaction Manager and access the *Account Administrator Guide* by selecting **Help > Lighthouse Transaction Manager Help > Account Management > Account Administrator Guide** from the menu.

Implementing i4Go

While i4Go implementation methods may vary, the use of i4Go is designed to be seamless whether the end user is entering their CHD on an e-commerce website or the end user is processing a customer's purchase at a kiosk.

Note: The term "end user" refers to the person who is entering information on the payment information form.

To implement i4Go, there are five key steps; each step requires developers to implement certain functionality to ensure the payment information is tokenized. The five steps are briefly outlined below and described in greater detail in the following subsections.

Requirement: The payment information must be tokenized for each transaction in order to benefit from reduced PCI DSS scope.

1. The purchase is initiated at the point of sale in a web browser-based environment. The end user's IP address is sent with the merchant's Access Token through the merchant's server to the i4Go server, thus requesting authorization* for Step 4.
2. The i4Go server returns an access block and the i4Go server address to the merchant's server. The merchant's server must modify the values of the i4m initialization parameters to include the access block and to post to the returned i4Go server address.
3. The CHD is entered on the payment information form (which is in an iFrame) and submitted directly to i4Go. i4Go sends the CHD to Lighthouse Transaction Manager where it is replaced with a TrueToken.

Tip: Swiping payment cards is supported. The end user simply clicks inside the iFrame and swipes the payment card using a point-to-point encryption (P2PE) enabled swipe reader or unencrypted magnetic swipe reader (MSR). Whether to support unencrypted MSRs can be configured. In addition, a script can be used to detect a card swipe without having to click inside the iFrame. For additional information, see the *Implementing Step 3* section.

WARNING! While i4Go's swipe collection logic supports both P2PE swipe readers and unencrypted swipe readers, Shift4 recommends always using P2PE swipe readers because the card data is encrypted inside the device, providing encryption from the point of swipe. These devices add another layer of security that unencrypted MSRs cannot provide. In addition, Shift4 recommends using P2PE devices with SRED as a function of the device, such as ID TECH SecuRED™ or SREDKey™.

4. The i4Go exit parameters are returned and mapped to the appropriate vendor-supplied form fields.
5. The application uses the TrueToken to process the transaction.

*This will attempt to authorize the end user's IP address to submit a single transaction through i4Go, regardless of where in the world the end user resides.

Implementing Step 1

Step 1: The purchase is initiated at the point of sale in a web browser-based environment. The end user's IP address is sent with the merchant's Access Token through the merchant's server to the i4Go server, thus requesting authorization for Step 4.

Please review these important details:

- This step is initiated from the merchant's Web server.
- This step requires an Access Token. (For additional information, see the *Prior to Implementation* section.)
- This step must be done for each tokenization attempt because the access block does expire.
- This step requires the use of `fuseaction=account.preauthorizeClient`, `i4go_clientip`, and `i4go_accesstoken` posted to https://access.shift4test.com (for certification) or https://access.i4go.com (for production). (The response will be JSON, not JSONP. For additional information, see the *i4Go Entry Parameters for the `preauthorizeClient` Request* section.)
  - *(Optional)* To include support for Apple Pay® and Google Pay™ wallets, this step requires the use of the `i4go_basket` container which has the nested `OrderDetails` container that has the nested `OrderNumber`, `Amount`, and `CurrencyCode` parameters. For additional requirements, see *Appendix D*.
  - *(Optional)* To return a signed JSON Web Token (JWT) to ensure data is not tampered with during the tokenization process, this step requires the use of `i4go_hs256key`.
  - *(Optional)* To have a MetaToken returned, this step requires the use of `i4go_metatoken`. (This feature has been deprecated and should not be implemented.)

Note: *Appendix D* provides guidelines and other information pertinent to implementing wallet support to an existing i4Go integration.

Requirement: Any call to https://access.shift4test.com or https://access.i4go.com must be direct posted and cannot be JSON or XML. In addition, developers must ensure the application retains a log of all authorization requests, including the client IP address, for troubleshooting purposes.

Sample Code – `preauthorizeClient` Request

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

Note: The `preauthorizeClient` request is a post from the merchant's Web server, which will be in their language of choice. The example below is in PHP, the details of which depend on the developer's environment. `$response` will contain the i4Go exit parameters.

```
$url = '[preauthorize client url]'; // https://access.i4go.com or https://access.shift4test.com
$ch = curl_init($url);


$data = array(
    'i4go_clientIP' => [end user's public ip address],
    'i4go_accessToken' => [merchant access token],
    'i4go_metatoken' => [F6 or IL based on type of MetaToken required; empty if not required. (This
feature has been deprecated and should not be implemented.)],
    'i4go_basket' => array(
        'OrderDetails' => array(
             'OrderNumber' => '[order identifier]',
             'Amount' => [amount is a currency total with no preceding currency symbol (e.g., $) and can
include up to two decimal places.],
             'CurrencyCode' => '[ISO-4217 currency code; optional]'
        )
    )
);


$post = array(serialize($data));


curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);


$response = curl_exec($ch);
curl_close($ch);
```

Implementing Step 2

Step 2: The i4Go server returns an access block and the i4Go server address to the merchant's server. The merchant's server must modify the values of the i4m initialization parameters to include the access block and to post to the returned i4Go server address.

Please review these important details:

- For a successful `preauthorizeClient` request, this step returns `i4go_response`, `i4go_responsecode`, `i4go_countrycode`, `i4go_accessblock`, and `i4go_server`.
  - The returned access block must be entered as the corresponding value to the `accessBlock` parameter.
  - The returned i4Go server address must be entered as the corresponding value to the `server` parameter.

Note: Modifying the values of the i4m initialization parameters is demonstrated in the *Sample Code - i4m Initialization and i4Go Exit Parameters* section.

- For a failed `preauthorizeClient` request, this step returns `i4go_response` and `i4go_responsecode`. (For additional information, see the *Accepted i4Go Exit Parameters* section.)

Implementing Step 3

Step 3: The CHD is entered on the payment information form (which is in an iFrame) and submitted directly to i4Go. i4Go sends the CHD to Lighthouse Transaction Manager where it is replaced with a TrueToken.

Please review these important details:

- This step is initiated from the end user's browser session.
- jQuery 3.x or greater is required.

Important: jQuery 2.x does not support Internet Explorer 6, 7, and 8. If support for those browsers is needed, you will need to implement i4Go using a method reviewed in the *i4Go Technical Reference Guide Using an Access Token* document. (Contact your Shift4 API support analyst for access to this document.)

- The i4m script must be in use, and it can be downloaded directly to the client from https://i4m.i4go.com/js/jquery.i4goTrueToken.js.
- *(Optional)* To support swiping payment cards when focus is anywhere on the page, the script must be in use and it can be downloaded directly to the client from https://i4m.i4go.com/js/jquery.cardswipe.js. If the script is not in use, then the end user must click inside the iFrame before swiping the payment card.
- To support unencrypted MSRs, the `encryptedOnlySwipe` parameter must be set to `false`. If `true`, then only P2PE swipe readers will be supported. (When set to `false`, unencrypted and encrypted MSRs are supported.)
- To only support card entry by use of an approved, secure device, set `deviceEntryOnly required` to `true`. By setting this to `true`, the card data will need to be captured on a card entry device (via dipping, swiping, manual entry, etc.) to continue with the transaction process. (A list of supported devices is on our website under hardware integrations: https://www.shift4.com/our-software-hardware-partners.) If `false`, manual entry of card data into the iFrame payment form will be allowed.

Tip: If you set `deviceEntryOnly required` to `true`, and you plan to require the card security code to be entered in the secure device, then you may want to set `cvv2Code visible` to `false`. This will allow the end user to bypass having to enter the card security code in the iFrame since it was already captured via the device.

- To support processing gift cards (including It's Your Card® gift cards) without requiring the end user to enter the card's expiration date, set the JavaScript flag `gcDisablesExpiration` to `true` and pass it to the iFrame. This will cause the Expiration field to be hidden in the iFrame when the end user selects Gift Card from the Payment Type list. If the flag is set to `false`, the Expiration Date field will be displayed in the iFrame regardless of which Payment Type is selected.
- To support processing gift cards (including It's Your Card gift cards) without requiring the end user to enter the card's security code, set the JavaScript flag `gcDisablesCVV2Code` to `true` and pass it to the iFrame. This will cause the Card Security Code field to be hidden in the iFrame when the end user selects Gift Card from the Payment Type list. If the flag is set to `false`, the Card Security Code field will be displayed in the iFrame regardless of which Payment Type is selected.
- There must be an empty `div` with an `id` of `i4goFrame`.

Tip: The page that calls the iFrame can specify a width, and then the content will stay within that width. If `frameAutoResize` is set to `true`, the iFrame will expand vertically as needed. And while the content within the iFrame is responsive, you will need to use the Bootstrap templates and make the iFrame itself responsive. The responsive calculations within the iFrame are based on the iFrame width.

- There must be a hidden object called `i4goTrueToken`, which is where all the configurations occur:
  - `server` – must be populated with the server address that is returned upon a successful `preauthorizeClient` request.
  - `accessBlock` – must be populated with the access block that is returned upon a successful `preauthorizeClient` request.
  - `self` – must be set to `document.location`.

- `template` – must be populated with the name of the template to be used (see *Appendix B* for additional information):
  - `bootstrap3`
  - `bootstrap3-horizontal`
  - `bootstrap4`
  - `bootstrap5`
  - `bootstrap5-labeled`
  - `shift4shop`
  - `plain`
  - `table`

WARNING! The Bootstrap 5 template is not supported with Internet Explorer 11 or earlier.

- `url` – must be populated with the appropriate i4m server address:
  - `https://i4m.shift4test.com` (for certification)
  - `https://i4m.i4go.com` (for production)

Note: Developers should only use `onSuccess` and `onFailure` or `formAutoSubmitOnSuccess` and `formAutoSubmitOnFailure`.

If the developer wants to control what happens when the tokenization request was successful and failed, `onSuccess` and `onFailure` should be used.

If the developer wants the merchant's payment information form (not the iFrame) to be automatically posted to the merchant's server (does not include any CHD), `formAutoSubmitOnSuccess` and `formAutoSubmitOnFailure` should be used.

The latter option is the simplest implementation for developers who are not JavaScript savvy.

- One of the following options:
  - `onSuccess` and `onFailure` – must configure events for success and failure if using client side event handling of the tokenization request.
  - `formAutoSubmitOnSuccess` and `formAutoSubmitOnFailure` – set to `true` to automatically post the merchant's payment information form (not the iFrame) to the merchant's server (does not include any CHD) if using server side event handling of tokenization events.

Requirement: If a timeout or null response is received, the process to tokenize the payment information must begin again at step 1.

- *(Optional)* `acceptedPayments` – set to "`AX,DC,GC,JC,MC,NS,VS`" by default, but can be changed to reflect the specific payment card types the merchant accepts. In addition, the default card types correspond to the available card types as of May 26, 2015 – they may increase or decrease over time.
- *(Optional)* `language` – set to the desired language. If not set, English is used.
  - "`en`" for English
  - "`es`" for Spanish
  - "`fr`" for French
  - "`pt`" for Portuguese
  - "`lt`" for Lithuanian

Tip: For an example, see https://myportal.shift4.com/index.cfm?action=development.i4mDemo.

- *(Optional)* `i4goInfo` – set to `false` by default to hide the i4Go text and logo. May be set to `true` to display the content, which is recommended by Shift4. (For additional information, see *Appendix B*.)
- *(Optional)* `cardNumber`, `expirationMonth`, and `expirationYear` fields have attributes that can be changed:
  - `label`
  - `placeholder`
  - `message`

Tip: The configured `label`, `placeholder`, and `message` text is displayed to the end user. To support a different language, change the configured text. In addition, you can modify `cssRules` to make the iFrame match your site's design.

- *(Optional)* `cardType`, `cvv2Code`, `cardholderName`, `streetAddress`, and `postalCode` fields also have attributes that can be changed:
  - `label`
  - `placeholder`
  - `message`
  - `visible`
  - `required`

Note: The `cardType` option does not have a `required` field. If `visible` is set to `true`, the field is required. For the `cvv2Code`, `cardholderName`, `streetAddress`, and `postalCode` options, if `visible` is set to `false`, the required setting is ignored.

- *(Optional)* The submit button (Secure My Payment Information) has the following attributes that can be changed:
  - `label`
  - `visible`

Note: If you hide the submit button by setting `visible` to `false`, then you will need the following call from the parent page to the iFrame to tokenize the information:

- `i4goTrueTokenObj`.submit();

Sample Code –`jQuery`, `i4m`, and `cardswipe` Scripts

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

```
<script src="[jQuery file]" type="text/javascript"></script>
<script src="https://i4m.i4go.com/js/jquery.i4goTrueToken.js" type="text/javascript"></script>
<script src="https://i4m.i4go.com/js/jquery.cardswipe.js" type="text/javascript"></script>
```

Sample Code – i4m Initialization and i4Go Exit Parameters

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

```
$(function(){
    $("#form-main").i4goTrueToken({
        server:      "",  // REQUIRED - get this from access.i4go.com call (or access.shift4test.com)
        accessBlock: "",  // REQUIRED - get this from access.i4go.com call (or access.shift4test.com)
        self:        "",  // REQUIRED - the full URL used to locate the current page
                          // MUST MATCH EXACTLY INCLUDING QUERY PARAMETERS - document.location should work


        template:   "plain", // The template you want. (Currently defaults to bootstrap3-horizontal. The
Bootstrap 5 template is not supported with Internet Explorer 11 or earlier.)
                             // Options:
                                // bootstrap3
                                // bootstrap3-horizontal
                                // bootstrap4
                                // bootstrap5
                                // bootstrap5-labeled
                                // shift4shop
                                // plain
                                // table
        gcDisablesExpiration: true, // Set to true to cause the Expiration field to be hidden in the
iFrame when the end user selects Gift Card from the Payment Type list. If the flag is set to false, the
Expiration field will be displayed in the iFrame regardless of which Payment Type is selected.
        gcDisablesCVV2Code: true, // Set to true to cause the Card Security Code field to be hidden in the
iFrame when the end user selects Gift Card from the Payment Type list. If the flag is set to false, the
Card Security Code field will be displayed in the iFrame regardless of which Payment Type is selected.
        encryptedOnlySwipe:     true,
        deviceEntryOnly:          { classes:"", label:"", required:false }, // If you set required to
true, the card data will need to be captured via dipping, swiping, manual entry, etc. on a secure and
approved card entry device to continue with the transaction process. If false, manual entry of card data
into the iFrame payment form will be allowed. To customize the message displayed to the end user, use
label.
        url:        "", // i4go server
                        // If not provided (the default), value will be calculated based on the "server"
parameter
                        // Defaults to production
                        // Must override for testing and certification to https://i4m.shift4test.com
```

```
        frameContainer:  "i4goFrame", // Only used if frameName does not exist
        frameName:       "",          // Auto-assigned if left empty
        frameAutoResize: true, // iframe will expand vertically as needed so content fits
        submitButton: {
             label: "Secure My Payment Information"
             visible: true
        }, // The text in quotes will be the text on the button. Button is hidden after submitting form.
Button can be visible or not; if not, then the following call is needed from the parent page to the iFrame
to tokenize the information: i4goTrueTokenObj.submit();
        frameClasses:    "",

        formAutoSubmitOnSuccess:  false,
        formAutoSubmitOnFailure:  false,

        onSuccess: function( form,data ){
            console.log("i4goTrueToken- onSuccess()",data);
        },
        onFailure: function( form,data ){
            console.warn("i4goTrueToken- onFailure()",data);
        },
        onComplete: function( form,data ) {
            console.log("i4goTrueToken- onComplete()",data);
        },

        //Wallet will asynchronously trigger individual callbacks.
        onWalletInit: function( wallet:string, enabled:boolean, reason:string ) {
            console.log("i4goTrueToken- onWalletInit()",wallet);
        },

        //This is used for wallet support. See the Dynamic Shipping and Sales Tax section in Appendix D
        for more information.
        onPaymentDataChanged: function( iobj, intermediatePaymentData, paymentDataRequestUpdate ){
             console.log("i4goTrueToken- onPaymentDataChanged()",intermediatePaymentData);
             return paymentDataRequestUpdate;
        },

        acceptedPayments: "AX,DC,GC,JC,MC,NS,VS",
        language: "en", // Set to desired language: en for English, es for Spanish, fr for French, pt for
        Portuguese, and lt for Lithuanian. If not set, English is used.
        // Auto populated form fields. Precedence: field name, field id
```

```
formPaymentResponse:            "customNameFori4go_response",
formPaymentResponseCode:        "customNameFori4go_responsecode",
formPaymentResponseText:        "customNameFori4go_responsetext",
formPaymentMaskedCard:          "customNameFori4go_maskedcard",
formPaymentToken:               "customNameFori4go_uniqueid",
formPaymentExpMonth:            "customNameFori4go_expirationmonth",
formPaymentExpYear:             "customNameFori4go_expirationyear",
formPaymentType:                "customNameFori4go_cardtype",
formCardholderName:             "customNameFori4go_cardholdername",
formStreetAddress:              "customNameFori4go_streetaddress",
formPostalCode:                 "customNameFori4go_postalcode",
formMetaToken:                  "customNameFori4go_metatoken",
formExtendedCardData:           "customNameFori4go_extendedcarddata",
formApplePayToken:              "customNameFori4go_applepaytoken",
formGooglePayToken:             "customNameFori4go_googlepaytoken",


// Advanced Options
/*
    payments: [
        { type: "VS", name: "Visa" },
        { type: "MC", name: "MasterCard" },
        { type: "AX", name: "American Express" },
        { type: "DC", name: "Diners Club" },
        { type: "NS", name: "Discover" },
        { type: "JC", name: "JCB" },
        { type: "GC", name: "Gift Card" }
    ],
    body:             { styles:{ "background-color": "##aaa", borderLeft: "5px solid ##ccc" } },
    label:            { classes:"control-label col-xs-4" },
    container:        { classes:"" },
    cardType:         { classes:"", label:"", placeholder:"", message:"", visible:true },
    // If visible, which is the default setting, it is required.
    cardNumber:       { classes:"", label:"", placeholder:"", message:"" },
    // Always visible and always required.
    expirationMonth:  { classes:"", label:"", placeholder:"", message:"" },
    // Always visible and always required.
    expirationYear:   { classes:"", label:"", placeholder:"", message:"" },
    // Always visible and always required.
```

```
            cvv2Code:         { classes:"", label:"", placeholder:"", message:"", visible:true,
required:true },

            // Can be visible or not; if visible, can be required or not. Default settings are visible and
required. If not visible, required setting is ignored. (If you set deviceEntryOnly required to true, and
you plan to require the card security code to be entered in the secure device, then you may want to set
cvv2Code visible to false. This will allow the end user to bypass having to enter the card security code
in the iFrame since it was already captured via the device.)

            cardholderName:   { classes:"", label:"", placeholder:"", message:"", visible:false,
required:true },

            // Can be visible or not; if visible, can be required or not. Default setting is not visible,
which means required setting is ignored.

            streetAddress:    { classes:"", label:"", placeholder:"", message:"", visible:false,
required:true },

            // Can be visible or not; if visible, can be required or not. Default setting is not visible,
which means required setting is ignored.

            postalCode:       { classes:"", label:"", placeholder:"", message:"", visible:false,
required:true },

            // Can be visible or not; if visible, can be required or not. Default setting is not visible,
which means required setting is ignored.

            i4goInfo:        { classes:"", label:"", visible:true, },

            // By default, set to false to hide the i4Go text and logo. May be set to true to display the
content, which is recommended by Shift4. (For additional information, see Appendix B.)

            cssRules: [

                "body{background-color: ##aaa;font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;}",

                "body{text-size: ""}"

                "label{color:##444;font-size:80%;font-weight:bold;}"

            ],


            // Debug flags: Simply creates additional console log messages

            // A debugger needs to be running to view. In no event does CHD get logged.

            debug:       false, // If true, displays console messages.

            remoteDebug: false  // If true, indicates width.

        */

    });

});
```

Sample Code – `i4goFrame` Div

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

```
<div id="i4goFrame" style="width: 500px;"></div> // specify width to keep content within it
```

Note: The `id` (i.e., `i4goFrame`) can be changed, but it must match the text that is configured in the `frameContainer` parameter.

Implementing Step 4

Step 4: The i4Go exit parameters are returned and mapped to the appropriate vendor-supplied form fields.

Please review these important details:

- The form fields have the default names displayed below:
  - `i4go_response`
  - `i4go_responsecode`
  - `i4go_responsetext`
  - `i4go_maskedcard`
  - `i4go_uniqueid`
  - `i4go_expirationmonth`
  - `i4go_expirationyear`
  - `i4go_cardtype`
  - `i4go_cardholdername`
  - `i4go_streetaddress`
  - `i4go_postalcode`
  - `i4go_metatoken`
  - `i4go_extendedcarddata`
  - `i4go_applepaytoken`
  - `i4go_googlepaytoken`
  - `i4go_signeddata`

Note: This is where the i4Go exit parameters are returned so they can be posted to the merchant's Web server. The form field names are a part of a hidden form and displayed in the *Sample Code - i4m Initialization and i4Go Exit Parameters* section. This section of code is only used if the developer would like to change the default names.

Important: Field names, as with most names in JavaScript, are case sensitive.

Sample Code – i4Go Exit Parameters

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

```
<form action="[merchant web server URL that processes the i4Go response, as per Implementing Step 5]"
method="post" style="display: none;">
    <input id="i4go_response" name="i4go_response" type="hidden" />
    <input id="i4go_responsecode" name="i4go_responsecode" type="hidden" />
    <input id="i4go_responsetext" name="i4go_responsetext" type="hidden" />
    <input id="i4go_cardtype" name="i4go_cardtype" type="hidden" />
    <input id="i4go_maskedcard" name="i4go_maskedcard" type="hidden" />
    <input id="i4go_uniqueid" name="i4go_uniqueid" type="hidden" />
    <input id="i4go_expirationmonth" name="i4go_expirationmonth" type="hidden" />
    <input id="i4go_expirationyear" name="i4go_expirationyear" type="hidden" />
    <input id="i4go_cardholdername" name="i4go_cardholdername" type="hidden" />
    <input id="i4go_streetaddress" name="i4go_streetaddress" type="hidden" />
    <input id="i4go_postalcode" name="i4go_postalcode" type="hidden" />
    <input id="i4go_metatoken" name="i4go_metatoken" type="hidden" />
    <input id="i4go_extendedcarddata" name="i4go_extendedcarddata" type="hidden" />
</form>
```

Implementing Step 5

Step 5: The application uses the TrueToken to process the transaction.

Note: The process is not over at this point because the TrueToken still needs to be authorized. For additional information on the authorization process, which uses Shift4's UTG, see RESTful API in MyPortal API Corner.

Appendix A – Accepted i4Go Parameters

Appendix A describes the accepted i4Go entry parameters developers will use for the `preauthorizeClient` request. This section also describes the accepted i4Go exit parameters that will be used to return data to the application.

Note: i4Go is not case sensitive with inbound parameter names. Outbound parameter names will always be in lowercase. For example, `i4go_uniqueid`.

Note: The Uniform Resource Identifier (URI) has a maximum limit of 2048 bytes in length.

i4Go Entry Parameters for the `preauthorizeClient` Request

Parameter	Valid Value	Required?	Description
`fuseaction`	• `account.preauthorizeClient` • Up to 255 bytes in length	**Yes**	Use the `fuseaction=account.preauthorizeClient` parameter to authorize the end user's IP address to submit a transaction through i4Go. When `fuseaction=account.preauthorizeClient` is in use, the `i4go_clientip` and `i4go_accesstoken` parameter must be used in conjunction.
`i4go_clientip`	• Numeric • xxx.xxx.xxx.xxx • Up to 255 bytes in length	**Yes**	Use the `i4go_clientip` parameter to post the end user's public IP address to i4Go. If the end user's IP address falls in the following ranges, then we substitute the requestor's IP address for the end user's IP address because all of these addresses are considered to be internal addresses. • 127.0.0.1/32 • 10.0.0.0/8 • 172.16.0.0/12 • 192.168.0.0/16

Parameter	Valid Value	Required?	Description
`i4go_accesstoken`	• String • Up to 255 bytes in length	**Yes**	Use the `i4go_accesstoken` parameter to post the merchant's Access Token to i4Go.
`i4go_metatoken`	• `F6` or `IL` • Up to 255 bytes in length	No	This feature has been deprecated and should not be implemented. Use the `i4go_metatoken` parameter to send the type of MetaToken required. • `F6` – Use when the last six digits of the MetaToken should be the first six digits of the payment card. • `IL` – Use when the last four digits of the MetaToken should be the last four digits of the payment card. If a MetaToken is not required, leave the `i4go_metatoken` parameter empty.
`i4go_basket`	• String	No	Use the `i4go_basket` container to include support for Apple Pay and Google Pay wallets. `i4go_basket` comprises a serialized JSON string containing the following: • `OrderDetails`
`i4go_hs256key`	• String • Up to 100 characters in length	No	Use the `i4go_hs256key` parameter to post the merchant-defined value to i4Go, thus returning a signed JWT to ensure the data is not tampered with during the tokenization process.

Order Details Object

Important: Use the following parameters to define the i4go_basket above for Apple Pay and Google Pay wallets support.

Parameter	Valid Value	Required?	Description
`OrderNumber`	• String (50)	**Yes**	This represents your Order Number or transaction identifier.
`Amount`	• Numeric (20)	**Yes**	Unformatted total transaction amount without any decimalization. Example: $100.00 = 10000, $123.67 = 12367, $.99 = 99
`CurrencyCode`	• String (3)	**Yes**	3 digit ISO 4217 value. Accepts either the Currency Number or Currency Code. Example: "840" or "USD"
`OrderDescription`	• String (256)	No	Brief Description of Items purchased.
`OrderChannel`	• String (16)	**Yes**	Specifies the order channel where the transaction was initiated. • M – MOTO (Mail Order Telephone Order) • R – Retail • S – eCommerce • P – Mobile Device • T – Tablet

Order Details Object Example

```
{

  "OrderDetails": {

    "OrderNumber": "",

    "Amount": "",

    "CurrencyCode": "",

    "OrderDescription": "",

    "OrderChannel": ""

  },


}
```

Accepted i4Go Exit Parameters

The accepted i4Go exit parameters returned in JSON format by i4Go are described and defined in the following table.

Parameter	Valid Value	When Returned?	Description
`i4go_response`	• `SUCCESS` or `FAILURE` • Up to 255 bytes in length	Always	The `i4go_response` parameter is used to return i4Go's Response Message.
`i4go_responsecode`	• Numeric list of one or more numbers • Up to 255 bytes in length	Always	The `i4go_responsecode` parameter is used to return all applicable i4Go Response Codes. For example: • `1` = SUCCESS • `301` = Not authorized For a complete list of Response Codes and Messages, please see *Appendix C*.
`i4go_countrycode`	• String • 2 bytes in length	Always	The `i4go_countrycode` parameter is used to return the two-character country code as assigned by iana.net and other Internet address authorities. For example: • `us` = United States • `??` = Unknown Note: International Organization for Standardization (ISO) Alpha-2 country codes are returned. Shift4 has seen at least two unofficial country codes (for example, `AP`) returned from Internet address authorities, which should be treated as unknown country codes.
`i4go_accessblock`	• String • Up to 1024 bytes in length	Successful `preauthorizeClient` Request	The `i4go_accessblock` parameter is used to return i4Go's access block.

Parameter	Valid Value	When Returned?	Description
i4go_server	• String • Up to 128 bytes in length	Successful preauthorizeClient Request	The i4go_server parameter is used to return the name of the i4Go server.
i4go_i4m_url	• String • Up to 128 bytes in length	Successful preauthorizeClient Request	The i4go_i4m_url parameter is used to return the name of the iFrame URL.
i4go_metatoken	• 16 numeric digits • Up to 20 bytes in length	Successful Tokenization Request	The i4go_metatoken parameter is used to return the MetaToken. If F6 was sent with the i4go_metatoken parameter, the last six digits of the MetaToken are the first six digits of the payment card. If IL was sent with the i4go_metatoken parameter, the last four digits of the MetaToken are the last four digits of the payment card.
i4go_responsetext	• String • Up to 255 bytes in length	Tokenization Request	The i4go_responsetext parameter is used to return a user friendly description that details why the request failed. If the request was successful, nothing is returned with the parameter.

Parameter	Valid Value	When Returned?	Description
i4go_maskedcard	• Asterisks and numeric • Up to 20 bytes in length	Successful Tokenization Request	The i4go_maskedcard parameter is used to return the masked payment card number. In addition, it can be used to display the information to the end user. For example, ********1119.
i4go_uniqueid	• String • 16 bytes in length	Successful Tokenization Request	The i4go_uniqueid parameter is used to return the Unique ID (TrueToken). The application will use and store the TrueToken to process the transaction.
i4go_expirationmonth†	• 1 or 2 Numeric Digits • Up to 2 bytes in length	Successful Tokenization Request	The i4go_expirationmonth parameter is used to return the expiration month of the payment card. For example, 04 - April is returned as 04.
i4go_expirationyear†	• 2 or 4 Numeric Digits • Up to 4 bytes in length	Successful Tokenization Request	The i4go_expirationyear parameter is used to return the expiration year of the payment card. For example, 2021 is returned as 2021.

Parameter	Valid Value	When Returned?	Description
i4go_cardtype†	• VS, MC, AX, DC, NS, JC, YC, GC, and PL • Up to 2 bytes in length	Successful Tokenization Request	The i4go_cardtype parameter is used to return the two-character code that identifies the payment card type being used. For example: • VS - Visa • MC - MasterCard • AX - American Express • DC - Diners Club/Carte Blanche • NS - Novus/Discover • JC - Japanese Credit Bureau (JCB) • YC - IT'S YOUR CARD • GC - Gift Card • PL - Private Label Payment Card
i4go_cardholdername†	• String • Up to 255 bytes in length	Successful Tokenization Request	The i4go_cardholdername parameter is used to return the name on the payment card, as entered by the end user, to i4Go. Or, when Apple Pay/Google Pay is used as the payment method on the transaction, then the billing/shipping name from the associated wallet will be returned.
i4go_postalcode†	• String • Up to 20 bytes in length	Successful Tokenization Request	The i4go_postalcode parameter is used to return the postal/ZIP code from the address that corresponds to the payment card, as entered by the end user, to i4Go.

Parameter	Valid Value	When Returned?	Description
i4go_streetaddress†	• String • Up to 50 bytes in length	Successful Tokenization Request	The i4go_streetaddress parameter is used to return the numerical portion of the street address that corresponds to the payment card, as entered by the end user, to i4Go.
i4go_extendedcarddata†	• String • Up to 8192 bytes in length	Successful Tokenization Request as Required	The i4go_extendedcarddata parameter is used to return extended card authentication data that corresponds to Apple Pay and Google Pay wallets support.
i4go_applepaytoken	• String • Up to 4096 bytes in length‡	Successful Tokenization Request and Apple Pay Was Used	This is optional and is a serialized JSON string representing the raw Apple Pay token that i4Go received. This token contains additional payment or cardholder information your application can leverage. This information is only returned if Apple Pay was used. For additional information, see Apple Pay documentation at: https://developer.apple.com/documentation/apple_pay_on_the_web/applepaypaymentmethodselectedevent/1778025-paymentmethod.
i4go_googlepaytoken	• String • Up to 4096 bytes in length‡	Successful Tokenization Request and Any Wallet Was Used	This is optional and is a serialized JSON string representing the raw Google Pay token (or a mimicked representation from an Apple Pay token) that i4Go received. This information is only returned if Google Pay or Apple Pay was used. For additional information, see Google Pay documentation at: https://developers.google.com/pay/api/web/reference/response-objects.

Parameter	Valid Value	When Returned?	Description
`i4go_signeddata`	• JWT	Successful Tokenization Request as Required	When `i4go_hs256key` is used, the `i4go_signeddata` parameter is used to return the corresponding JWT. (The JWT's signature can then be verified before attempting to submit the transaction. For additional information on JWT, see https://jwt.io/.)

†i4Go is designed to keep real and sensitive CHD information out of the merchant's Web server or hosting provider's system. If you send this information to the merchant's systems with the primary account number (PAN), you are defeating i4Go's purpose. The information may be sent as long as the PAN is not included.

‡A maximum length of 4096 bytes should be safe; however, this is not controlled by Shift4.

Appendix B – Example of i4Go in an iFrame

To render an example of i4Go in an iFrame, click here and then click on the available templates, options, and languages.

Note: The example displays what is possible. Based on defaults or changes you make to the code, some fields may or may not be displayed. In addition, you can change the width using `i4goFrame`.

WARNING! The Bootstrap 5 template is not supported with Internet Explorer 11 or earlier.

Appendix C – i4Go Response Codes and Messages

Appendix C describes i4Go Response Codes and Messages an end user may see.

Developers can display the Response Message returned by i4Go, or they can process the Response Code and display a customized response message.

The following table describes each Response Code and the corresponding Response Message end users will see (if the developer chose to display the Response Message rather than customizing their own) in the event of a success or an error during the `authorizeClient` or tokenization request process.

Response Code	Response Message	Explanation
1	SUCCESS	The i4Go request was successful.
86	FAILURE	The i4Go request was not successful because the user clicked cancel on the i4Go payment information form. (This response is deprecated.)
`authorizeClient` Function Errors		
300	Server does not accept `authorizeClient` function	The `authorizeClient` function failed because the i4Go server neither requires nor accepts the `authorizeClient` function.

Response Code	Response Message	Explanation
301	Not authorized	One of the following errors has occurred with the required and hidden parameter fields on the payment information form: • The i4Go server does not have the value submitted with the `i4go_clientip` parameter in queue. • The required i4Go Entry Parameters and the `i4go_accessblock` parameter were not returned to the specified i4Go server name. • The value submitted with the `i4go_accessblock` parameter is invalid. • The allotted time for using the corresponding values to the `i4go_clientip` and `i4go_accessblock` parameters has expired.
302	`authorizeClient` Error: Please try again.	Unable to complete the `authorizeClient` request.
302	`authorizeClient` Error: Communication error - #cfhttp.statusCode#. Please try again.	A Shift4 communication service is not responding.
302	`authorizeClient` Error: Invalid XML response - no children. Please try again.	A Shift4 communication service is responding in an incorrect format.
302	`authorizeClient` Error #val(xmlDoc.response.xmlAttributes.code)#: Please try again.	A Shift4 communication service responded with an error.
302	`authorizeClient` Error: No notes returned. Please try again.	A Shift4 communication service is not returning expected data.
302	`authorizeClient` Error: No server returned. Please try again.	A Shift4 communication service is not returning expected data.

Response Code	Response Message	Explanation
503	USAGE ERROR: i4go_accessToken is required (or i4Go_accountId and i4Go_siteId)	Valid credentials, either an AccessToken or an AccountID and SiteID, were not supplied.
503	USAGE ERROR: Invalid i4go_accessToken format	The supplied AccessToken is improperly formatted.
505	USAGE ERROR: i4go_clientIP is required	The ClientIP address was not supplied.
505	USAGE ERROR: Invalid i4go_clientIP format (IPv4 format required)	The supplied ClientIP address is improperly formatted.
Configuration Errors		
	**Note:** Configuration Errors alert the developer to problems during the development and approval process. The end user should never see these messages.	
503	USAGE ERROR: i4Go_AccessToken is required	The developer must modify the payment information form to include the merchant's Access Token. For additional information, see the *Prior to Implementation* section.
504	CONFIGURATION ISSUE: Access block invalid or missing	The value submitted with the `i4go_accessblock` parameter field is invalid (due to length) or missing. The developer must contact Shift4 to enable this function. Once enabled, i4Go returns an access block when a successful `authorizeClient` request has been received. The developer must modify the payment information form to return this value with the `i4go_accessblock` parameter (which includes the merchant's Access Token).
505	USAGE ERROR: Invalid i4Go_ClientIP format (IPv4 format required)	The value submitted with the `i4go_clientip` parameter field is invalid.

Response Code	Response Message	Explanation
506	USAGE ERROR: i4Go_ClientIP is required	The developer must modify the payment information form to post the end user's IP address to i4Go.
Informational Messages		
601	i4Go down for maintenance - try again later	The i4Go system is down for maintenance.
602	Lighthouse Transaction Manager down for maintenance - try again later	Lighthouse Transaction Manager is temporarily unavailable due to maintenance.
603	Lighthouse Transaction Manager merchant database down for maintenance - try again later	Lighthouse Transaction Manager is operational, but the merchant database is temporarily unavailable due to maintenance.
Blacklist Errors		
9827	Country [country code] is blacklisted	The end user is attempting to process a transaction from a country that has been blacklisted for security reasons. i4Go will not allow connections from this country.
9828	IP Address [IP address] is blacklisted	The end user is attempting to process a transaction from an IP address that has been blacklisted for security reasons. i4Go will not allow connections from this IP address.

Appendix D – Implementing Apple Pay and Google Pay Wallets

Wallets provide cardholders a friendly, consistent, and simple user experience.

Before implementing wallets to an i4Go integrated application, we STRONGLY recommend you start with a working i4Go integration first. Adding wallet support to an existing i4Go integration is very easy and the changes are detailed here.

Apple Pay

What is Apple Pay?

Apple Pay is easy and works with the Apple devices you use every day. You can make contactless, secure purchases in stores, in apps, and on the web. And you can send and receive money from friends and family right in Messages. Apple Pay is a safer way to pay, and even simpler than using your physical card.

Apple Pay Prerequisites

Before adding Apple Pay to your application, you must read, understand, follow, and accept the Apple Pay on the Web: Acceptable Use Guidelines. These guidelines are published and mandated by Apple and are outside the control of Shift4.

Apple Pay does require proof of domain ownership prior to assigning credentials. This process is detailed in the *Step 1 – Account Setup* section below.

For more information, see Apple Pay on the Web Overview.

Google Pay

What is Google Pay?

Google Pay brings together all the ways you can pay with Google.

Enter your card information once and use it to:

- Tap and pay to make purchases with your phone (see country and device availability).
- Buy items in apps and on websites (see country availability).
- Fill in forms automatically on Chrome.
- Buy Google products.
- Send money to friends and family (US only).

You can also use your gift cards, loyalty cards, tickets, and coupons with Google Pay when you shop at your favorite stores.

Google Pay Prerequisites

Before adding Google Pay to your application, you must review and adhere to the following Google Pay API Terms of Service, Acceptable Use Policy, and Brand Guidelines. These guidelines are published and mandated by Google and are outside the control of Shift4.

In addition, the following documents can be used for reference: Google Pay Web Developer Documentation, Google Pay Web Integration Checklist, and Google Pay for Payments Overview.

Implementing Wallets

Now that you have completed your i4Go integration for tokenizing cardholder data, it is time to add wallet support. The wallets, Apple Pay and Google Pay, will allow for a much quicker and streamlined user experience, along with adding an additional security layer for your customers.

This section assumes you have an understanding of the i4Go API and a working i4Go integration. Only details for implementing wallet support to your interface are included below.

Step 1 – Account Setup

Before requesting Apple Pay and/or Google Pay setup and credentials, go through and thoroughly read and understand the prerequisites detailed previously.

Shift4 is in the process of making the setup process and the request for development/testing and production credentials self-serve, but that is under development. Until this is available, you will need to speak directly to your Shift4 representative for account setup and credentials. Please check back for further updates as this is a work in progress.

Apple Pay does require proof of domain ownership prior to assigning credentials. Before Apple Pay credentials can be assigned, you will be provided a file from Shift4 and instructions on where to put them on your server (located here). The file is for proof of domain ownership and does not contain any sensitive information. The file should remain on the server(s) even after ownership confirmation.

Step 2 – CSS and JavaScript – Adding Additional Includes

i4Go wallet support required the addition of two Shift4 include files: a style sheet (CSS) and a JavaScript (JS) file. They can be downloaded directly to the client from:

- https://i4m.i4go.com/css/wallets.css
- https://i4m.i4go.com/js/wallets.js

In addition, for Google Pay support, a Google JavaScript (JS) file is required:

- https://pay.google.com/gp/p/js/pay.js

Sample Code

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

```
<link rel="stylesheet" type="text/css" href="https://i4m.i4go.com/css/wallets.css">
<script src="https://i4m.i4go.com/js/wallets.js" type="text/javascript"></script>
<script src="https://pay.google.com/gp/p/js/pay.js" type="text/javascript"></script>
```

Step 3 - i4GoTrueToken Initiation

When including wallet support, we have added the following:

- `onWalletInit: function( wallet:string, enabled:boolean, reason:string )` – This is optional. Wallet will asynchronously trigger individual callbacks.
- `i4go_extendedcarddata` – **This is required.** You must receive the extended card data and pass it to your Shift4 payment request. Failure to pass the information will result in authorization failures and settlement downgrades (higher rates). The result is supplied into the `extendedcarddata` field of the Shift4 API.
- `i4go_applepaytoken` – This is optional and is a serialized JSON string representing the raw Apple Pay token that i4Go received. This token contains additional payment or cardholder information your application can leverage. This information is only returned if Apple Pay was used. For additional information, see Apple Pay documentation at: https://developer.apple.com/documentation/apple_pay_on_the_web/applepaypaymentmethodselectedevent/1778025-paymentmethod.
- `i4go_googleppaytoken` – This is optional and is a serialized JSON string representing the raw Google Pay token (or a mimicked representation from an Apple Pay token) that i4Go received. This information is only returned if Google Pay or Apple Pay was used. For additional information, see Google Pay documentation at: https://developers.google.com/pay/api/web/reference/response-objects.

Sample Code

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

```
…
// Wallet event
onWalletInit: function(wallet:string, enabled:boolean, reason:string ) {
        console.log("i4goTrueToken- onWalletInit()",wallet);
},
// Auto populated form fields. Precedence: field name, field id
formPaymentResponse:          "customNameFori4go_response",
formPaymentResponseCode:      "customNameFori4go_responsecode",
formPaymentResponseText:      "customNameFori4go_responsetext",
formPaymentMaskedCard:        "customNameFori4go_maskedcard",
formPaymentToken:             "customNameFori4go_uniqueid",
formPaymentExpMonth:          "customNameFori4go_expirationmonth",
formPaymentExpYear:           "customNameFori4go_expirationyear",
formPaymentType:              "customNameFori4go_cardtype",
formCardholderName:           "customNameFori4go_cardholdername",
formStreetAddress:            "customNameFori4go_streetaddress",
formPostalCode:               "customNameFori4go_postalcode",
formMetaToken:                "customNameFori4go_metatoken",
formExtendedCardData:         "customNameFori4go_extendedcarddata",
formApplePayToken:            "customNameFori4go_applepaytoken",
formGooglePayToken:           "customNameFori4go_googlepaytoken",
…
```

Step 4 – HTML – Adding Buttons

We're to the easiest part. Simply include one or both buttons in your HTML code. The buttons are hidden by default and will only reveal themselves if the associated wallet is available. Feel free to add your own style to the buttons, just be sure you refer to the branding guidelines (see the *Apple Pay Prerequisites* or *Google Pay Prerequisites* section).

Sample Code

WARNING! The sample code was designed for demonstration purposes only. While you can use this code as a template, additional code will most likely need to be added to make this sample "production ready."

```
<button class="pay-button pay-hidden apple-pay-button"></button>
<button class="pay-button pay-hidden google-pay-button"></button>
```

Step 5 – Test

Test your integration to ensure it works.

Step 6 – Advanced Features

For most of the advanced wallet features, i4Go uses Google Pay as the standard. Meaning, if you code for Google Pay (with very few or no exceptions) you will receive Apple Pay compatibility.

We chose this path because in our view, Google Pay seems to have more developer friendly documentation. Also, developers have a wider selection of developer and debugging tools whereas Apple Pay is limited to tools running in iOS only.

Google Pay Button Styles

Google Pay controls the button styles via JavaScript settings, which are passed through our i4goTrueToken settings.

```
wallet:   {
        // For the following button attributes, see Google documentation. Default settings used when left empty.
        buttonColor:        "",     // default: A Google-selected default value. Currently black but it may change over time (default).
                                    // black: A black button suitable for use on white or light backgrounds.
                                    // white: A white button suitable for use on colorful backgrounds.
        buttonType:         "",     // buy: "Buy with Google Pay" button (default).
                                    // donate: "Donate with Google Pay" button.
                                    // plain: Google Pay button without additional text.
                                    // translated button label may appear if a language specified in the viewer's browser matches an available language.
        buttonSizeMode:     "",     // static: Button has a static width and height (default).
                                    // fill: Button size changes to fill the size of its container.
        buttonRootNode:     ""      // HTMLDocument or ShadowRoot
}
```

For additional information, see Google Pay documentation at:
https://developers.google.com/pay/api/web/reference/request-objects#ButtonOptions.

Apple Pay Button Styles

Apple Pay controls the button styles via CSS classes, which you add directly to the button tag. For additional information, see Apple Pay documentation at:
https://developer.apple.com/documentation/apple_pay_on_the_web/styling_the_apple_pay_button_with_css.

Name, Address, and Static Shipping Options

The wallets can optionally return information from the cardholder, including: name, email address, phone number, and shipping address. Any or all of these fields can be required. Due to various local, state, and federal personally identifiable information (PII) laws, it is recommended that you only require what you need to fulfill your order. In other words, if you are not shipping goods, do not require shipping information as this would simply increase your liability scope for PII.

```
wallet:  {
         nameRequired:           null,       // true/false/null - null uses addressRequired
         addressRequired:        false,
         emailRequired:          false,
         phoneNumberRequired:    false,
         allowedCountryCodes:    [],
         shippingOptionRequired: false,
         shippingOptions:        [],         // first option will be the default
         shippingType:           "shipping"  // shipping, delivery, storePickup, servicePickup
}
```

Dynamic Shipping and Sales Tax

Dynamic shipping and sales tax support is an extension of the prior *Name, Address, and Static Shipping Options* section. The difference between static and dynamic is that dynamic requires an additional event to be defined in order to support this feature. The feature allows for shipping options to change based on different shipping addresses being selected and provided by the cardholder within the wallet.

For example, if the cardholder selects to use their home address, one set of shipping options is made available. If the cardholder changes the shipping address to a work address or a hotel address, a different set of shipping options are displayed. In both these cases, the appropriate sales tax is applied to the order.

Tip: Before adding dynamic shipping and sales tax options, get the prior *Name, Address, and Static Shipping Options* section working first.

```
wallet:                     {
        nameRequired:       null,                   // true/false/null - null
uses addressRequired.
        addressRequired:    false,
        emailRequired:      false,
        phoneNumberRequired: false,
        allowedCountryCodes: [],
        shippingOptionRequired: false,
        shippingOptions:    [],                         // first option will
be the default.
        shippingType:       "shipping"              // shipping, delivery,
storePickup, servicePickup
}
```

Appendix E – Additional Resources

Appendix E contains a list of additional resources that may be helpful and provide additional guidance. Direct any specific questions about these documents to their respective publishers.

Product Support

For assistance with this and any other Shift4 product, visit https://www.shift4.com/support.

Live Support

Information about troubleshooting techniques and handling special problems that may occur during installation, configuration, or use can be obtained by contacting the Shift4 Support department at 888.857.9751, option 2.

Development/Integration Support

For assistance handling problems that may occur when developing an application programming interface (API) and integrating Shift4 products with your system, contact your Shift4 API support analyst.

Feedback

Your feedback regarding Shift4 products and documentation is welcome, encouraged, and we appreciate your comments. If you have any documentation comments or suggestions about this guide, please send them to us at techdocs@shift4.com.

Shift4 Guides and Documentation

The following Shift4 Guides and Documentation may provide additional helpful information and are located here:

- *Securing the Merchant's Site That Uses i4Go*
- *Account Administrator Guide*
- *Card Manufacturing Tips*

Industry Websites

The following site provides additional industry guidelines and standards:

- https://www.pcisecuritystandards.org