CHASE PAY

E-commerce Implementation Specification

December 2017

Document Version 3.41

Chase Pay® E-commerce Implementation Specification

Version 3.41

December 2017

What's New

Changes from previous version 3.31 and 3.4 are the following:

- **About this Specification** section is re-written
- **References** sub-section is re-written
- **Getting Help** sub-section is re-written
- **Payment Flows** are replaced
- **Chapter 2** is completely re-written
- **API Attributes** are updated
- **Error Handling Table** is inserted at end of Chapter 4

Table of Contents

About This Specification

This specification provides steps on implementing Chase Pay in an E-Commerce environment.

NOTE: Provide any feedback on this document to your assigned Chase Account Executive.

Audience

This specification is intended for use by Merchants and internal Chase support personnel assisting in a Chase Pay implementation effort in an E-Commerce environment. Merchants using this specification must also have an acquiring relationship with Chase Merchant Services.

References

Use the **Chase Pay – CMCP Documents Index** provided by your Chase Account Executive or Chase Technical Implementations Manager for a list of documents needed depending on the type of Chase Pay Implementation.

Getting Help

A Chase Pay Technical Implementations Manager is available to assist Merchants in any Chase Pay implementation. Consult with your Chase Account Executive for more information.

Abbreviations & Terminology

The following list clarifies several terms and conventions used throughout this specification.

- **Chase Account Executive**
  This person is the Merchant's advocate within Chase. Acts as primary contact point for Merchant.

- **Chase Merchant Services**
  Also referred to as "Chase" in this document. Chase facilitates the flow between the Merchant host and the Chase Pay Services host.

- **Chase Pay Services Host**
  Chase Pay backend platform responsible for Chase Pay authentication, account management, and payment details.

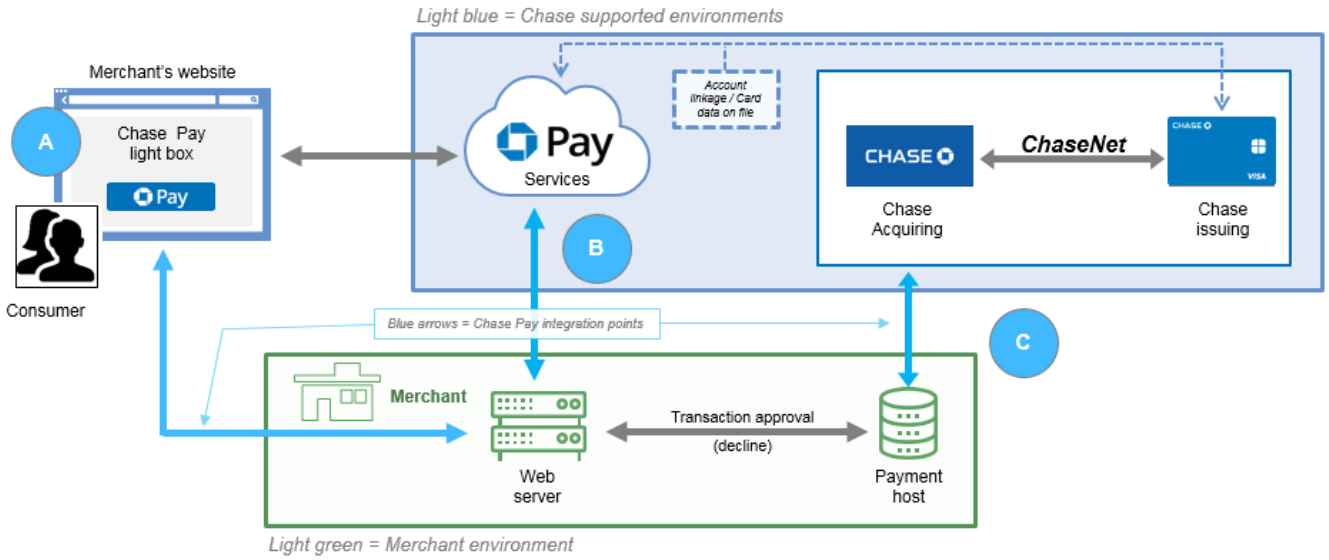
- **Chase Technical Implementations Manager**
  This person (or team) supports the Merchant with Chase Pay implementations. Duties include coordinating digital wallet integrations and running the certification process.

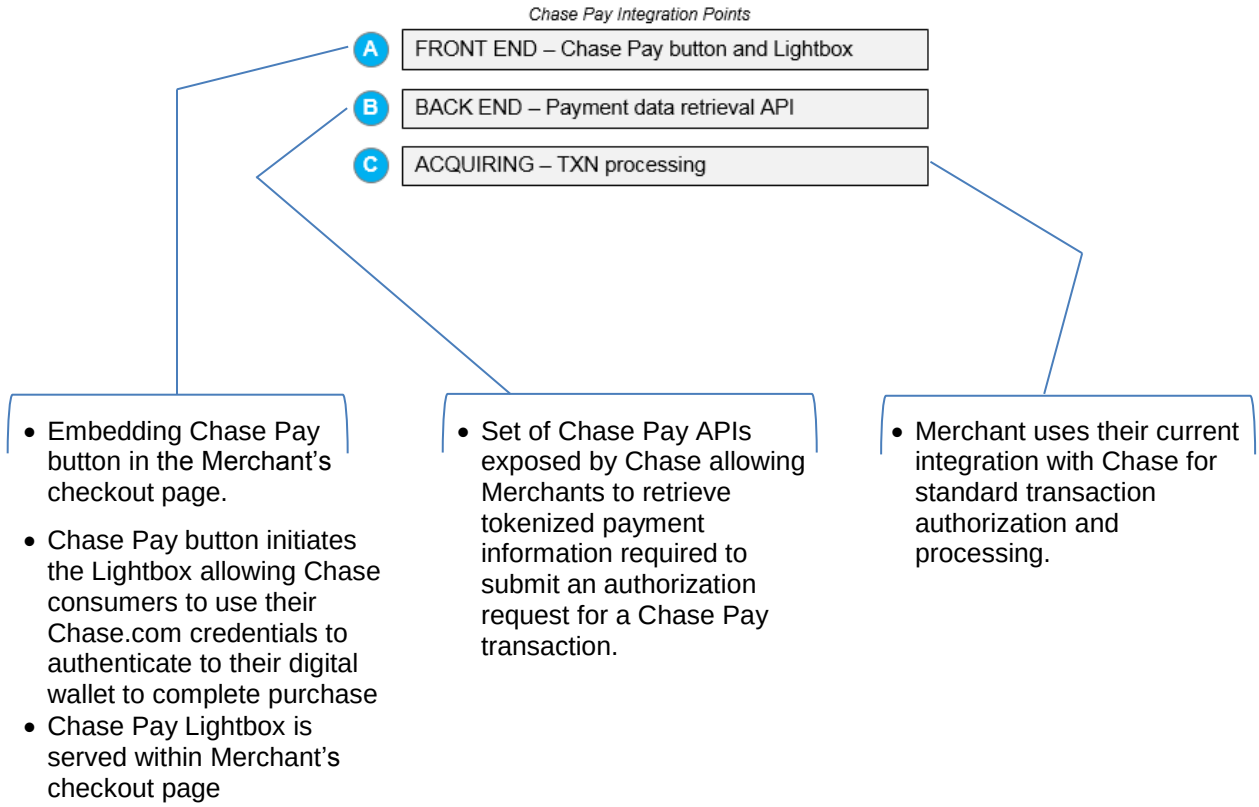
- **E-commerce**
  The purchase of goods and services from an online Merchant website or digital storefront typically performed from within a browser (desktop or mobile).

- **Merchant Host**
  Merchant backend end system that has the responsibility to interface to Chase / Chase Pay Services.

Chapter 1 Getting Started

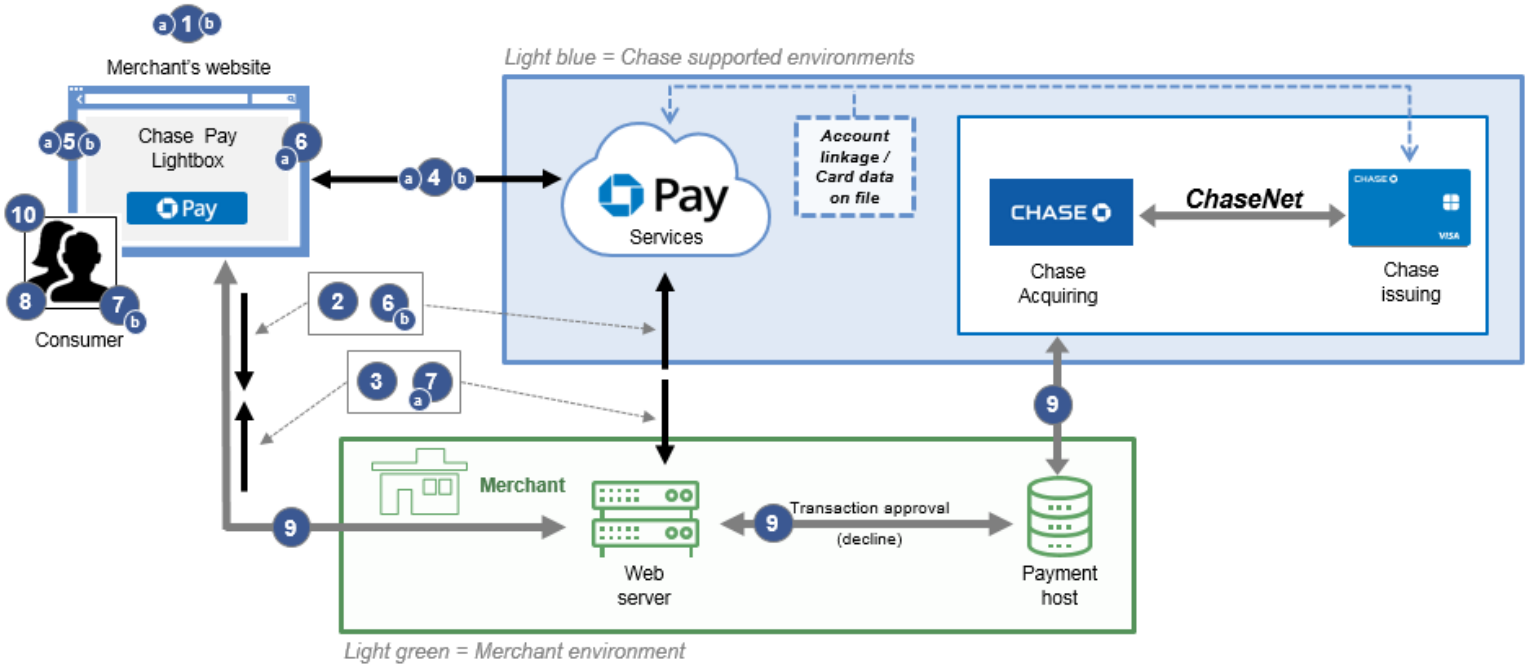
A typical Chase Pay E-commerce transaction consists of 3 primary interfaces and integration steps:





Chase Pay E-commerce Process flow

The following diagram provides a high level flow of an E-commerce transaction using Chase Pay.



1	Consumer clicks on Chase Pay button on Merchant's website to initiate checkout. (a) Chase Pay Button is rendered on Merchant's check out page; (b) button click initiates *startcheckout* process.
2	Browser initiates *MerchantSession* request to Chase Pay Services (CPS).
3	CPS returns a Digital Session ID (DSID) to Merchant.
4	DISID is (a) is sent to CPS for validation and (b) CPS validates against open payment session.
5	Lightbox appears on top of Merchant's website, (a) consumer authenticates with their Chase.com credentials, and (b) consumer selects payment options from digital wallet.
6	(a) Lightbox closes and *GetCheckOutData* API request is initiated. (b) DSID is passed to CPS via Merchant.
7	CPS responds to API (a) with payment details which (b) Merchant presents for confirmation by consumer
8	Payment instrument and details are confirmed by consumer on checkout page.
9	Transaction is routed to ChaseNet for BAU approval through Merchant's processing interface.
10	Upon approval of transaction, final response communicating sale is displayed to consumer.

Before Proceeding

Tokenization is key

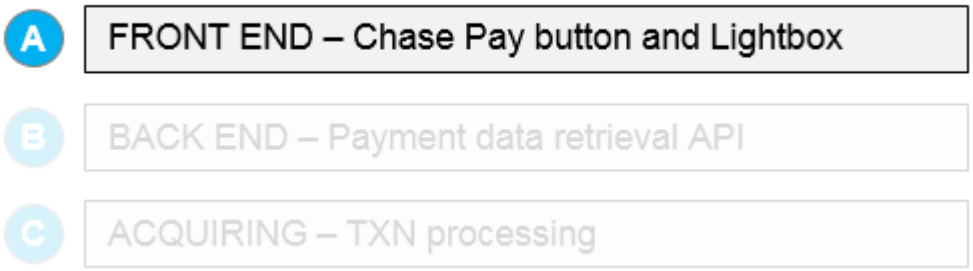
- All Chase Pay transactions are tokenized based on EMVCo standards.
- As tokenization is a key step in any Chase Pay payment processing, Chase Pay requires Merchants to support EMVCo tokenized transaction processing including passing the Token Requestor ID in both the authorization and clearing transaction requests.

- The Token Requestor ID will be returned to the Merchant along with the payment information details following a successful consumer Lightbox interaction.
- The transaction cryptogram must be placed in the existing cardholder Account Authentication Value (AAV) field of the authorization request.

Branding guidelines

Merchants must follow Chase Pay branding guidelines (button color, shape, and graphics). Summary button brand guidelines are provided in the **Chase Pay Merchant Implementation Guide**.

Chase Pay Integration Points



Chapter 2 Wallet Frontend Integration

Introduction

This section explains how to integrate the Chase Pay Lightbox into a new or existing checkout flow on the Merchant's payment page. Integration of the front-end portion of Chase Pay consists of 5 steps:

1. Including the Chase Pay JavaScript in any page Merchant wishes to display Chase Pay as a payment option, but not in pages where Chase Pay will not be displayed.
2. Registering JavaScript callbacks for Chase Pay events.
3. Configuring Chase Pay for Merchant's need.
4. Inserting Chase Pay buttons or branding elements onto payment page.
5. Inserting the "Learn More" informational link.

Step 1: Add the Chase Pay JavaScript Element

The Chase Pay script element specifies the source of the chase.com hosted JavaScript file which provides the Chase Pay JavaScript API. The file's size is kept very small to make as little impact as possible on page load performance. Merchant should also make sure not to include it on pages where Chase Pay will not be presented to consumers. This creates unnecessary load on Chase servers and will be an unnecessary HTTP request for consumers.

```
<script
src="https://pwcpsb.chase.com/pwc/checkout/js/v20170521/list.action?type=raw&applId=PWC&channelId=CWC&version=1"></script>
```

JavaScript URLs

During development and certification, Merchant can use the certification URL below. Once the certification has been completed by Chase Merchant Services and Chase Pay has been released into production, Merchant will have to change the script URL to the production URL as indicated below.

Certification	https://pwcpsb.chase.com/pwc/checkout/js/v20170521/list.action?type=raw&applId=PWC&channelId=CWC&version=1
Production	https://pwc.chase.com/pwc/checkout/js/v20170521/list.action?type=raw&applId=PWC&channelId=CWC&version=1

API Implementation Options

The API is exposed one of two ways depending on the technology used on Merchant's site. If RequireJS is used, a named AMD module is exposed. If Merchant does not use RequireJS, the API is available in a global object.

RequireJS

```
require(['JPMC_ChasePay_Checkout'], function (ChasePay) {
    // Access API methods here with the ChasePay argument
})
```

Global object

```
// Access API methods in this global variable
window.JPMC.ChasePay
```

Step 2: Register JavaScript callbacks for Chase Pay events

Registering callbacks for predefined events allows Chase to notify Merchant of the following:

- **ChasePay.EventType.START_CHECKOUT** indicates the Chase Pay button has been clicked
- **ChasePay.EventType.COMPLETE_CHECKOUT** indicates the Chase Pay Lightbox session has completed successfully
- **ChasePay.EventType.CANCEL_CHECKOUT** indicates the Chase Pay Lightbox session has been canceled by the customer

Instructions based on the EventType:

ChasePay.EventType.START_CHECKOUT

- This EventType indicates the Chase Pay button has been clicked. Merchant will register a callback for this event if Merchant is using a clickable Chase Pay button, inserted with ChasePay.insertButtons().
- Merchant should skip this callback registration if radio buttons, a select dropdown, or some other method of mutually exclusive selection is being used.
- When Chase calls the Merchant's event callback with this eventType, Merchant must make a call to their server via a Merchant's web service call. This web service should call Chase's SOAP/XML MerchantSession API to create a Digital Session ID and it should not be cacheable.
- Merchant uses the response header "Cache-Control: no-cache, no-store" to prevent caching of web service
- Based on the response returned from Merchant's service, call should either be ChasePay.startCheckout() or ChasePay.showError(). If ChasePay.startCheckout() is not called within 10 seconds after Merchant's callback is invoked, Chase will automatically invoke ChasePay.showError() to display an error to the consumer.
- While Merchant is making the call to their server to retrieve the session ID, Chase will display a darkened overlay and a loading animation to signal to the consumer that the Chase Pay button is working as intended. If Session ID cannot be retrieved, animation and overlay are programmed to end after a fixed time period and an error message will appear.

```
JPMC.ChasePay.on( JPMC.ChasePay.EventType.START_CHECKOUT, function () {

    /**
    Get a Digital Session ID from your API. You must get a new Digital Session ID
    for every startCheckout event! Make sure this API endpoint cannot be cached by
    using the response header: cache-control: no-cache, no-store. We display an
    error after 10 seconds, so this request should timeout when that happens
    **/
    jQuery.ajax({
        url: 'https://yourDomain.com/path/to/chasepay/session/creation/api',
        method: 'POST',
        timeout: 10000
    }).done( function successHandler (digitalSessionId) {

        /**
        Pass in the Digital Session ID to start the Chase Pay lightbox flow
        **/
        JPMC.ChasePay.startCheckout(digitalSessionId);

    }).fail( function failureHandler () {

        /**
        Show an error in the Chase Pay lightbox to tell the customer an
        error occurred
        **/
        JPMC.ChasePay.showError();
    });
});
```

ChasePay.EventType.COMPLETE_CHECKOUT

- This EventType confirms that the consumer has completed their payment selection.
- From here, Merchant proceeds to retrieve the elements (payment details) of the payment selected through the GetCheckoutDetails API via the Merchant's backend server. Payment details are then displayed to the consumer to confirm the purchase.
- Chase will pass in the Merchant's API callback a params object containing the Digital Session ID which was originally passed in as params.sessionToken.

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

```
JPMC.ChasePay.on( JPMC.ChasePay.EventType.COMPLETE_CHECKOUT, function (params) {

    /**
    The Chase Pay lightbox session has been completed. The params.sessionToken
    is populated with the original Digital Session ID that was passed in
    Note: this method name is for example purposes only
    **/
    continueToNextStepInTheFlowAfterChasePay(params.sessionToken);

});
```

ChasePay.EventType.CANCEL_CHECKOUT

- This eventType confirms that the consumer cancelled the Chase Pay Lightbox experience.
- No action is required by Chase when this event is triggered as it is for information purposes only.

```
JPMC.ChasePay.on( JPMC.ChasePay.EventType.CANCEL_CHECKOUT, function (params) {

    /**
    The Chase Pay lightbox session has been cancelled by the user.
    Chase doesn't require any action to be taken here by you,
    this is purely informational
    **/

});
```

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

Step 3: Configure Lightbox Functionality

The ChasePay.configure() method provides several options to customize the functionality of the Chase Pay Lightbox. The following keys are used in the dictionary object passed to this method:

Key	Value	Usage
Language	String ("en" or "es")	The language that will be displayed in the Chase Pay Lightbox • This must be an ISO 639-1 language string • Currently only English ("en") and Spanish ("es") are supported • The default value is "en"
zIndex	Number	The CSS z-index value for the Chase Pay Lightbox • The Lightbox should always be the top-most element on Merchant's page when the Lightbox is open • Ads, offers, etc. must not be displayed on top of the Lightbox • The default value is 999999
sessionWarningTime	Number	The amount of time (in milliseconds) before a session timeout warning is displayed to the consumer • Only values lower than the default value will be honored • The default value is 340000 (5m 40s)
sessionTimeoutTime	Number	The amount of time (in milliseconds) before the Lightbox is automatically closed due to session timeout • Only values lower than the default value will be honored • The default value is 400000 (6m 40s)

To configure these options, Merchant should call ChasePay.configure() and pass in a dictionary object. The keys of this object must exactly match the keys defined in this section. Merchant only needs to pass the keys they want to change, and this method can be called as often as necessary prior to launching the Chase Pay Lightbox.

```
JPMC.ChasePay.configure({
    language: "es",
    zIndex: 100001,
    sessionWarningTime: 300000,
    sessionTimeoutTime: 360000
});
```

Step 4: Insert and Customize Chase Pay Buttons or Branding Elements

Implementation Options

The Chase Pay Lightbox experience can be initiated through a clickable button or through a single option select element like a radio button or dropdown menu. For single option select implementations, the Chase Pay image will not be clickable. Below we will refer to clickable buttons as “buttons” and non-clickable images as “branding elements”.

Different methods are used to implement these two options (detailed examples follow below):

- For clickable buttons use ChasePay.insertButtons().
- For branding elements, use ChasePay.insertBrandings().

Option 1: Inserting a Clickable Button

First, Merchant will need a container element to insert the Chase Pay button into:

```
<div id="chasePayButton"></div>
```

Merchant can then call the ChasePay.insertButtons() method, ensuring that Chase Pay is available prior to inserting the buttons:

```
if ( JPMC.ChasePay.isChasePayUp ) {
    JPMC.ChasePay.insertButtons({
        containers: ['chasePayButton']
    });
}
```

NOTE: Merchant must use ChasePay.isChasePayUp boolean to determine whether or not to insert Chase Pay buttons and branding elements. If there is any content on Merchant’s web site related to Chase Pay, Merchant must also use this boolean to hide it.

If Merchant wants to insert multiple buttons, each additional element ID should be added as another item in the container array. If Merchant wants to pass the HTML element objects themselves, this can also be done:

```
/**
Let's assume you have this HTML:
<div id="chasePayButton1"></div>
<div id="chasePayButton2"></div>

Then you can either pass strings:
**/
if ( JPMC.ChasePay.isChasePayUp ) {
    JPMC.ChasePay.insertButtons({
        containers: ['chasePayButton1', 'chasePayButton2']
    });
}
/**
Or HTML Element objects:
**/
var chasePayElement1 = document.getElementById('chasePayButton1');
var chasePayElement2 = document.getElementById('chasePayButton2');


if ( JPMC.ChasePay.isChasePayUp ) {
    JPMC.ChasePay.insertButtons({
        containers: [chasePayElement1, chasePayElement2]
    });
}
```

NOTE: For additional configuration options, consult the “Chase Pay Iconography Customization Options” section that follows Option 2.

Option 2: Inserting a Branding Element (mutually exclusive selection option)

If Merchant’s other payment methods are selected by the consumer with radio buttons, a select dropdown, or some other method of mutually exclusive selection, then Merchant will want to insert Chase Pay Branding elements into Merchant’s HTML to communicate to consumers the option to use Chase Pay. Chase will not attach event handlers to branding elements. This means that the Merchant will not receive a START_CHECKOUT event when a branding element is clicked. The process for inserting Chase Pay Branding elements is nearly identical to inserting clickable Chase Pay Buttons.

First, Merchant needs a container element:

```
<li>
    <label for="chasePayRadio">
        <input id="chasePayRadio" type="radio">
        <span id="chasePayBranding"></span>
    </label>
</li>
```

Merchant can then insert a branding element using the ChasePay.insertBrandings() method. Any combination of element IDs or HTML element objects are accepted in the containers array.

```
if ( JPMC.ChasePay.isChasePayUp ) {
    JPMC.ChasePay.insertBrandings({
        containers: ['chasePayBranding']
    });
}
```

NOTE: For additional configuration options, consult the “Chase Pay Iconography Customization Options” section that follows this section.

Chase Pay Iconography Customization Options

Both of these methods take the same argument, a dictionary object that defines how the button or branding element should look (dimensions and color) and where to insert the button or branding element. The following keys are used in the dictionary object passed in this method:

Key	Value	Usage
Color	String	This field controls the color of the graphic. Either 'blue' or 'white' is accepted. See branding guidelines for acceptable use. "blue" will result in a blue graphic with a white Chase Pay mark - this is the default "white" will result in a white graphic, with a blue and black Chase Pay mark
Containers	Array	This field must be an Array containing any combination of HTML element ID strings or instances of HTML element objects. Chase will insert buttons or branding elements as children of these elements, depending on the method called.
Height	Number	This field controls the height of the inserted Chase Pay graphic. The default height is 44 pixels. If a different height is desired, pass the desired height in pixels as a number in this field. The minimum is 34 and the maximum is 94.

Key	Value	Usage
Width	Number	This field controls the width of the inserted Chase Pay graphic. Only use this field when a static-width button is desired. If your site follows Responsive Web Design principles, it's recommended to leave this field blank, resulting in a button that takes up 100% of the width of the container element, allowing the button to also follow Responsive Web Design principles and Merchant site's CSS. If a number is provided, a minimum is enforced for every given height. The default height of 44 has a minimum width of 100 pixels.

Examples:
For a responsive-width blue button/image with a height of 40 pixels, use:

```
{
    containers: ['myChasePayButtonContainerID'],
    height: 40
}
```

The resulting graphic will look like this in a 300-pixel-wide container:

The same graphic when the container's size is reduced to 175 pixels will look like this:

For a 200x36 static-size white button/image, use:

```
{
    containers: ['myChasePayButtonContainerID'],
    color: 'white',
    height: 36,
    width: 200
}
```

The resulting graphic will look like this on a dark background:

For a small, badge-like button/image, use:

```
{
    containers: ['myChasePayButtonContainerID'],
    height: 34,
    width: 77
}
```

The resulting graphic will look like this:

Implementing Chase Pay as a Mutually Exclusive Option

When using a mutually-exclusive selection method in combination with Chase Pay branding elements, Merchant will not register a callback for the START_CHECKOUT event. Instead, Merchant will be in control of the sequence of events:

1. Determine when the user selects Chase Pay as their payment method
2. Invoke the Chase Pay loading animation
3. Get a Digital Session ID from Merchant's web service, which retrieves the ID from the MerchantSession SOAP/XML service that is not cacheable.
4. Pass the Digital Session ID to ChasePay.startCheckout()

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

```
/**
In this example, this function is called when
a customer has indicated they would like to use Chase Pay with
a radio button, select dropdown, or some other mutually-exclusive
selection method and have taken some action, like clicking
a "Checkout" button, that indicates they are ready to complete
their purchase.
**/
function startChasePayCheckout() {

    /**
    This call shows the Chase loading animation to let customers know
    their selection was recorded and that work as started to fulfill
    their request
    **/
    JPMC.ChasePay.showLoadingAnimation();

    /**
    Get a Digital Session ID from your API. You must get a new Digital Session ID
    for every startCheckout event! Make sure this API endpoint cannot be cached by
    using the response header: cache-control: no-cache, no-store. We display an
    error after 10 seconds, so this request should timeout when that happens
    **/
    jQuery.ajax({
        url: 'https://yourDomain.com/path/to/chasepay/session/creation/api',
        method: 'POST',
        timeout: 10000
    }).done( function successHandler (digitalSessionId) {

        /**
        Pass in the Digital Session ID to start the Chase Pay lightbox flow
        **/
        JPMC.ChasePay.startCheckout(digitalSessionId);

    }).fail( function failureHandler () {

        /**
        Show an error in the Chase Pay lightbox to tell the customer an
        error occurred
        **/
        JPMC.ChasePay.showError();

    });
}
```

Chase Pay Status and Availability

While Chase strives to make Chase Pay available 24/7, there may be circumstances where Chase Pay is down or unavailable. When this is the case, Chase will always give Merchant an accurate status in the
'isChasePayUp' Boolean.

isChasePayUp example

```
/**
When Chase Pay is up, insert buttons or branding elements.
These IDs are for example purposes only, these are whatever you choose.
**/
if ( JPMC.ChasePay.isChasePayUp ) {
    JPMC.ChasePay.insertButtons({ containers: ['chasePayButton'] });
    JPMC.ChasePay.insertBrandings({ containers: ['chasePayBranding'] });

/**
When Chase Pay is not up, hide custom selection methods and Chase Pay
related content. These IDs are for example purposes only, these are
whatever you choose.
**/
} else {
    document.getElementById('chasePayRadioButton').style.display = 'none';
    document.getElementById('chasePayContent').style.display = 'none';
}
```

NOTE: Merchant must use ChasePay.isChasePayUp boolean to determine whether or not to insert Chase Pay buttons and branding elements. If Merchant has content on their site related to Chase Pay, boolean must be used to hide it.

Step 5: Insert the “Learn More” Link

Chase exposes the getSupplementalContent() method to provide Chase Pay related content messaging. See below for an example of a simple learn more link that uses Chase Pay supplemental content:

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

Supplemental Content

```
<a href="#" id="chase-pay-learn-more-link">Learn More</a>
<div id="chase-pay-learn-more" style="display: none;"></div>
<script>
$("#chase-pay-learn-more-link").click(function(){
    window.JPMC.ChasePay.getSupplementalContent(function (content){
        var heading = "<h3>"+ content.supplemental_heading + "</h3>";
        var body = "<p>" + content.supplemental_body + "</p>";

        var $learnMoreContent = $("#chase-pay-learn-more");
        $learnMoreContent.html(heading + body);
        $learnMoreContent.css("display", "block");
    });
})
</script>
```

API

Method or Property	Usage
`on({EventType} eventType, {Function} callback)`	Call this method once for each EventType you want to listen for. The first argument must be from the EventType enumeration, and the second argument must be a function.

`configure({Object} dictionary)`	Call this method to configure the lightbox options. The dictionary object doesn't need to contain all keys, just the options you want to change. The options are:
	• The language displayed in the Chase Pay lightbox
	  • This must be an ISO 639-1 language string
	  • Currently only English ("en") and Spanish ("es") are supported
	• The z-index of the lightbox on your page
	  • The lightbox should always be the topmost element on your page while it's open
	  • Ads, offers, etc. must not be displayed on top of the lightbox
	• When we display a session timeout warning to the customer that their session will time out soon
	  • Only values lower than the default value will be honored
	  • This must be a Number in milliseconds
	• When we timeout the session and close the lightbox if the user took no action after the warning
	  • Only values lower than the default value will be honored
	  • This must be a Number in milliseconds
	Dictionary format (defaults shown):
	`{ language: "en", zIndex: 999999, sessionWarningTime: 340000, // 5m 40s sessionTimeoutTime: 400000 // 6m 40s }`

`showLoadingAnimation()`	Call this method once you've determined the customer intends to use Chase Pay and you've started the process of retrieving a Digital Session ID. This will display a Chase branded loading animation, giving the customer an indication things are working. You have 10 seconds to retrieve a **new** Digital Session ID before an error is automatically shown to the customer.
`startCheckout({String} sessionToken)`	Launches the Chase Pay lightbox. The Digital Session ID is required. The Digital Session ID should be generated immediately before being passed to this method. showLoadingAnimation() should be called before this method while you fetch the Digital Session ID.
`showError()`	Call this method if you've already called showLoadingAnimation() and have encountered an issue retrieving a Digital Session ID. This will show an error message to the customer to avoid a confusing experience.

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

`insertButtons({Object} buttonConfig)`	Call this method to insert Chase Pay buttons after script-load. Buttons emit the START_CHECKOUT event when clicked.
	
	The configuration object passed only requires the containers Array; other fields are optional and based on your specific needs. The fields are:
	
	• The containers in which to insert Chase Pay buttons
	  • The array can contain HTML Element ID strings and/or HTML Element objects
	• The color of the button as a String
	  • The default is "blue", the only other currently supported option is "white"
	• The height of the button, as a Number
	  • The default is 44 pixels.
	  • The minimum is 34 pixels
	  • The maximum is 94 pixels
	• The width of the button, as a Number
	  • The default is 100%, so the button always takes up 100% of the width of the parent container. This is the recommended option.
	  • The minimum width changes based on the height, and is 100 pixels for the default height of 44 pixels
	
	Dictionary format (examples shown):
	
	`{ containers: ['myChasePayButtonContainerID', myChasePayButtonContainerElement], color: 'blue', height: 40, width: 200 }`

`insertBrandings({Object} brandingConfig)`	Call this method to insert Chase Pay buttons after script-load. Branding elements do not emit the START_CHECKOUT event. The configuration object follows the same pattern as the insertButtons() configuration object.
`setLanguage({String} language)`	Call this method to set the language used in the Chase Pay Lightbox. English and Spanish are currently supported. The language parameter must be ISO 639-1 compliant, e.g. "en" for English or "es" for Spanish. The default language for the Chase Pay Lightbox is English. It is recommended to call this API whenever a user updates the language on your site.
`getSupplementalContent({Function} handleContent(content) { ... })`	Call this method to get Chase Pay related messaging. The handleContent callback function will be invoked with a single argument of an object containing the following keys: • `supplemental_heading` - a heading for an explanation of Chase Pay • `supplemental_body` - an explanation of Chase Pay
`Boolean isChasePayUp`	This Boolean indicates whether Chase Pay is available. Chase Pay may be unavailable for any reason. If this Boolean is false, hide any Chase Pay related features or content.
`Enum EventType`	Enumeration of the following event types: • `START_CHECKOUT` - Fired when a Chase Pay Button has been clicked • `COMPLETE_CHECKOUT` - Fired when the Chase Pay Lightbox session has ended successfully • `CANCEL_CHECKOUT` - Fired when the Chase Pay Lightbox session has ended unsuccessfully

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

Payment Completion

Once the Lightbox has invoked the COMPLETE_CHECKOUT event callback, Merchant can then request for the payment details selected by the consumer by submitting a getCheckoutDataRequest message to the Chase SOAP/XML service.

NOTE: The Digital Session ID that was used to establish the consumer's session is required for all subsequent requests. The session has an expiration window of 30 minutes. Merchant can generate standard authorization requests, using the payment details provided in the response, to Merchant's existing acquiring interface with Chase.

Putting it All Together: Full Examples

Option 1: Clickable Button

```
<!DOCTYPE html>
<html>
<head>
    <title>Chase Pay Clickable Button Example</title>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>

    <!-- Chase Pay Script -->

    <script
     src="https://pwcpsb.chase.com/pwc/checkout/js/v20140921/list.action?type=ra
     w&applId=PWC&channelId=CWC&version=1"></script>

    <script>
        // Register startCheckout callback
        JPMC.ChasePay.on( JPMC.ChasePay.EventType.START_CHECKOUT, function () {

            // Get a Digital Session ID from your API. You must get a new
              Digital Session ID
            // for every startCheckout event! Make sure this API endpoint
              cannot be cached by
            // using the response header: cache-control: no-cache, no-store
            jQuery.ajax({
                url:
                   'https://yourDomain.com/path/to/chasepay/session/creation/api
                   ',
                method: 'POST',
                timeout: 10000 // We display an error after 10 seconds, so this
                   request should timeout when that happens
            }).done( function successHandler (digitalSessionId) {



                // Pass in the Digital Session ID to start the Chase Pay
                   lightbox flow
                JPMC.ChasePay.startCheckout(digitalSessionId);
```

```
        }).fail( function failureHandler () {

            // Show an error in the Chase Pay lightbox to tell the customer
               an error occurred
            JPMC.ChasePay.showError();

        });
    });

    // Register completeCheckout callback
    JPMC.ChasePay.on( JPMC.ChasePay.EventType.COMPLETE_CHECKOUT, function
     (params) {
        // The lightbox session has been completed. The params.sessionToken
          is
        // populated with the original Digital Session ID
          continueToNextStepInTheFlowAfterChasePay(params.sessionToken);
    });

    // Register cancelCheckout callback
     JPMC.ChasePay.on( JPMC.ChasePay.EventType.CANCEL_CHECKOUT, function
     (params) {
        // This is purely informational
    });


    // Set any optional configurations needed for Chase Pay. Uncomment and
     modify these as
     // needed by your application, or omit this call if no configurations
     are necessary.
    JPMC.ChasePay.configure({
        // language: 'es',
        // zIndex: 100001,
        // sessionWarningTime: 300000,
        // sessionTimeoutTime: 360000
    });

     Function continueToNextStepInTheFlowAfterChasePay(digitalSessionId) {
     // Call your backend to retrieve the payment details from the
    // Chase Paymentech GetCheckoutData SOAP/XML service and display
    // a screen to the user to confirm their order as usual
    };

    if ( JPMC.ChasePay.isChasePayUp ) {
        JPMC.ChasePay.insertButtons({
            containers: ['chasePayButton']
        });
    }

    $(document).ready(function(){
        $("#chase-pay-learn-more-link").click(function(event){
            event.preventDefault();

            // Request the supplemental content passing in a callback that
               display the content to the customer
```

```
                    window.JPMC.ChasePay.getSupplementalContent(function(content)
                    {
                         var heading = "<h3>" + content.supplemental_heading +
                         "</h3>";
                      var body = "<p>" + content.supplemental_body + "</p>";

                      var $learnMoreContent = $("#chase-pay-learn- more");
                      $learnMoreContent.html(heading + body);
                      $learnMoreContent.css("display", "block")
                 });
             })
         })
    </script>
</head>
<body>
    <div id="chasePayButton"></div>
    <a href="#" id="chase-pay-learn-more-link">Learn More</a>
    <div id="chase-pay-learn-more" style="display: none;"></div>
</body>
</html>
```

Option 2: Radio Button

```
<!DOCTYPE html>
<html>
<head>
    <title>Chase Pay Radio Button Example</title>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>

     <!-- Chase Pay Script -->
    <script
     src="https://pwcpsb.chase.com/pwc/checkout/js/v20140921/list.action?type=r
     aw&applId=PWC&channelId=CWC&version=1"></script>

    <script>
        // Register completeCheckout callback
       JPMC.ChasePay.on( JPMC.ChasePay.EventType.COMPLETE_CHECKOUT, function
        (params) {
            // The lightbox session has been completed. The params.sessionToken
              is
            // populated with the original Digital Session ID
              continueToNextStepInTheFlowAfterChasePay(params.sessionToken);
        });

        // Register cancelCheckout callback
        JPMC.ChasePay.on( JPMC.ChasePay.EventType.CANCEL_CHECKOUT, function
         (params) {
            // Merchants dont usually do anything here as there is no need
        });
```

```
// Set any optional configurations needed for Chase Pay. Uncomment and
 modify these as
 // needed by your application, or omit this call if no configurations
 are necessary.
JPMC.ChasePay.configure({
    // language: 'es',
    // zIndex: 100001,
    // sessionWarningTime: 300000,
    // sessionTimeoutTime: 360000
});

window.continueToNextStepInTheFlowAfterChasePay = function
 (digitalSessionId) {
    // Call your backend to retrieve the payment details from the
    // Paymentech Chase Pay GetCheckoutData SOAP/XML service and
      display
    // a screen to the user to confirm their order as normal
};

// Event handler for the form submit event
function onChoosePaymentMethod(event) {
    // Stop the form element from submitting
    event.preventDefault();

    var chasePayRadio = document.getElementById('chasePayRadio');

    // Chase Pay was selected by the user
    if (chasePayRadio.checked) {

        // Show the Chase Pay loading animation
        JPMC.ChasePay.showLoadingAnimation();

        // Start a Chase Pay session from your backend
        jQuery.ajax({
            url:
               'https://yourDomain.com/path/to/chasepay/session/creation
               /api',
            method: 'POST',
            timeout: 10000 // We display an error after 10 seconds, so
               this request should timeout when that happens
        }).done( function successHandler (digitalSessionId) {

            // Pass in the Digital Session ID to start the Chase Pay
                lightbox flow
            JPMC.ChasePay.startCheckout(digitalSessionId);

        }).fail( function failureHandler () {

            // Show an error in the Chase Pay lightbox to tell the
                customer an error occurred
            JPMC.ChasePay.showError();

        });
    }
}
```

```
        $(document).ready(function () {
            $('#paymentMethod').on('submit', onChoosePaymentMethod);

            if ( JPMC.ChasePay.isChasePayUp ) {
                // If Chase Pay is up, insert the Chase Pay branding
                JPMC.ChasePay.insertBranding({
                    containers: ['chasePayRadioLabel']
                });
                // Show the radio button by removing the default style of
                  display:none
                document.getElementById('chasePayOption').style.display = '';
            }

              // Request the supplemental content passing in a callback that
              display the content to the customer
            $("#chase-pay-learn-more-link").click(function(event){
                event.preventDefault();

                   window.JPMC.ChasePay.getSupplementalContent(function(content)
                   {
                     var heading = "<h3>" + content.supplemental_heading +
                        "</h3>";
                     var body = "<p>" + content.supplemental_body + "</p>";
                     var $learnMoreContent = $("#chase-pay-learn-more");
                     $learnMoreContent.html(heading + body);
                     $learnMoreContent.css("display", "block");
                });
            })
        });
    </script>

</head>
<body>
    <form id="paymentMethod">
        <ul id="paymentRadios">

            <li id="chasePayOption" style="display:none">
                <label for="chasePayRadio">
                    <input type="radio" name="paymentMethod" value="chasePay"
                       id="chasePayRadio"  />
                    <div id="chasePayRadioLabel"></div>
                    <a href="#" id="chase-pay-learn-more-link">Learn More</a>
                    <div id="chase-pay-learn-more" style="display:
                       none;"></div>
                </label>
            </li>

            <li>
                <label for="creditCardRadio">
                    <input type="radio" name="paymentMethod" value="creditCard"
                       id="creditCardRadio" />
                    Credit Card
                </label>
```

```
            </li>

        </ul>
        <button type="submit">Continue</button>
    </form>

</body>
</html>
```

Integration Review:

By now Merchant should have completed the following:

- ***Embedding of the Chase Pay script in the checkout page***
- ***Inserting the Chase Pay Option into the payment form***
- ***Coding to initiate the Chase Pay Lightbox and capture the various Chase Pay events***
- ***Enable checking the availability of the Chase Pay service***
- ***Posting of relevant payload information to Merchant's backend server***

Merchant also has the Chase Pay button URL information for both the certification and production.

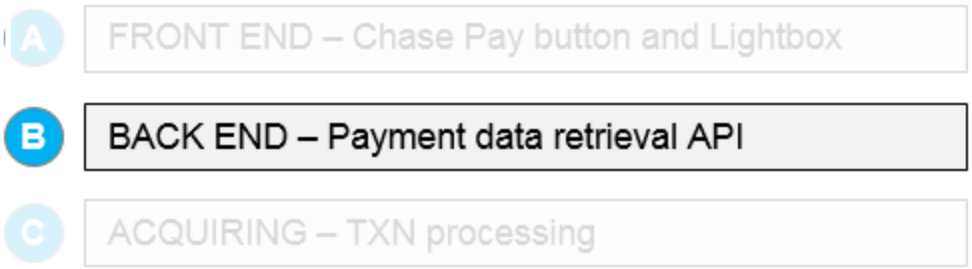
Next Steps:

All items that Merchant needs to consider before commencing the backend wallet integration are described in the next chapters

Considerations:

- How to display the button image correctly
  - Merchant does not need to worry about rendering the image as the button script will put together the markup to render the button to Merchant's page.
- Where to get more information about jQuery.ajax() method
  - Refer http://api.jquery.com/jQuery.ajax/

Chase Pay Integration Points



Chapter 3 Wallet Backend Integration

Getting Started

The following base items will be needed as part of the Chase Pay integration:

Protocol	Security	Connectivity	Credentials	Failover

Comments:

- Chase supports **SOAP or XML** interfaces
- The communication protocol is **HTTPS**
- **Non-authenticated SSL sessions and TLS 1.2** is used
- **POST requests only** are supported
- **Web Services** is used, and the WSDL and XSD information is available to you for both certification and production
- **HTTP/1.0 and HTTP/1.1 MIME header** are supported
- Merchant must conform to our **connectivity, authentication,** and **failover handling** procedures as explained in this chapter
- **Connectivity, authentication,** and **failover handling** with Chase Pay services is done through Chase' s Orbital Gateway

Communication Protocol

- HTTPS is the only communication protocol supported by Chase Pay.
- This method provides a single-threaded (or synchronous) model in which a Merchant makes an HTTPS request to the service and then blocks the request until the service sends back the HTTPS response.
- A single Merchant interface can initiate multiple HTTPS requests at once.

SSL Implementation

- TLS 1.2
- Chase Pay Service interfaces must be accessed using HTTPS.
- Requires the use of an SSL implementation.
- The service uses a non-authenticated SSL session, meaning the Merchant is not authenticated using a digital certificate as a component of the SSL negotiation.
- Non-SSL postings should never be made across a network that is external or not totally controlled and secure.
- If a clear text request is made to Chase Pay Service URLs, the service will reject the request, returning a SOAP faultstring value of 20403 or XML ProcStatus 20403 error.

Posting to a URL

- Chase Pay Service will only provide responses to HTTP POST requests.
- GET requests are not supported.
- This method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URL in the Request-Line.
- HTTP version number: 1.0 or 1.1.
- The Request URL for the Chase Pay Service will always be:
    - /DigitalWalletService for SOAP
    - /DigitalWalletXmlService for XML services

Interfaces

Merchant can choose the Chase Pay interface of their preference. Chase Pay Service offers SOAP and XML as interface options.

Web Services

- The Web Service interface supports version 1.1 of the W3C SOAP (www.w3.org/TR/SOAP).
- Developers can use the programming language they are most comfortable with, whether it is C# (.NET), Java, C++, or others.
- Developers should be familiar with SOAP, Web Service Definition Language (WSDL), and other standards that are part of Web Services.
- WSDL explains how to get the service, parameters, and service responses in an XML document that can be read or machine-processed.
- The WSDL is only accessible to servers that are registered. However, the WSDL files are available for viewing from any server at the following address:

https://secure.paymentech.com/developercenter/catalog/pages/chasepay/documents

Chase Pay Service WSDL files - Certification

https://wsvar1.chasepaymentech.com/DigitalWallet/wsdl/DigitalWalletService.wsdl

Chase Pay Service WSDL files – Production

https://ws1.chasepaymentech.com/DigitalWallet/wsdl/DigitalWalletService.wsdl

SOAP Message Style and Encoding

The Chase Pay uses Document/Literal, rather than RPC/Encoded, SOAP messaging.

XML Service

- The current XSD (XML Schema Definition) published for interfacing with the Chase Pay Service is PTI0.6.
- There are separate request and response XSDs: DigitalWalletRequest_PTI06.xsd and DigitalWalletResponse_PTI06.xsd.
- This allows Chase requests and responses to be easily interpreted and manipulated using the W3C (World Wide Web Consortium) DOM (Document Object Model) or SAX (Simple API for XML) APIs.

MIME Header

The XML interface requires a MIME header Chase Pay Service supports both the HTTP/1.0 and HTTP/1.1 MIME (Multipurpose Internet Mail Extensions) Header specifications for describing the message payload along with other information that allows it to process the incoming transaction request, and the outgoing reply.

The table below describes the Request MIME-Header elements:

Element Name	Description	Required
MIME-Version	Must be 1.0 or 1.1.	M
Content-type	Defines the XML version number Current version*: application/PTI06 *Make sure to check the latest XML version number at time of the implementation	M
Content-length	This value defines the length of the Request XML Document.	M
Content-transfer-encoding	Defines the encoding of the associated XML document. Recommended encoding is text.	M
Request-number	Must always be 1.	M
Document-type	Defines whether this is a Request or Response. This value must always be Request.	M
Interface-Version	Optional MIME-Header element that can be used by Chase Paymentech in production support.	O

M = Mandatory, O = Optional

Common to both SOAP and XML Service

- Errors or unexpected behaviors can result if any characters in the request payload do not match the character encoding specified in the request.
- Most messages specify “UTF-8” encoding and contain ASCII characters.
- The Chase Pay Service also supports ISO-8859-1 encoding, often referred to as the Latin-1 character set. Messages containing French, Spanish, or other Latin rooted languages may require ISO-8859-1 encoding instead.

Connectivity

Connectivity to Chase Pay Services is provided through Chase Pay Orbital Gateway using

- Internet Connection, or
- MPLS / VPN connection

There are different URLs and IPs associated with XML and SOAP, and these vary by connection type: internet vs. MPLS.

Internet Connection

XML Production, Port 443

Primary https://orbital1.chasepaymentech.com/DigitalWalletXmlService
Secondary https://orbital2.chasepaymentech.com/DigitalWalletXmlService

XML Certification, Port 443

Primary https://orbitalvar1.chasepaymentech.com/DigitalWalletXmlService
Secondary https://orbitalvar2.chasepaymentech.com/DigitalWalletXmlService

SOAP Production, Port 443

Primary https://ws1.chasepaymentech.com/DigitalWalletService
Secondary https://ws2.chasepaymentech.com/DigitalWalletService

SOAP Certification Port 443

Primary https://wsvar1.chasepaymentech.com/DigitalWalletService
Secondary https://wsvar2.chasepaymentech.com/DigitalWalletService

MPLS/VPN Connection

XML Chase Pay Production, Port 443

Primary 206.253.184.96/DigitalWalletXMLService
Secondary 206.253.180.97/DigitalWalletXMLService

XML Chase Pay Certification, Port 443

Primary 206.253.180.98/DigitalWalletXMLService
Secondary 206.253.184.98/DigitalWalletXMLService

SOAP Chase Pay Production Port 443

Primary 206.253.184.161/DigitalWalletService
Secondary 206.253.180.162/DigitalWalletService

SOAP Chase Pay Certification, Port 443

Primary 206.253.184.163/DigitalWalletService
Secondary 206.253.180.163/DigitalWalletService

NOTE: For Merchants connecting over MPLS, it is not possible to conduct a DNS lookup to validate the certificate against the IP as the DNS response will resolve to the external IP rather than the MPLS IP provided to the Merchant.

NOTE: IP authentication via Internet is available for existing Orbital Gateway Merchants who are already using this option.

Managing Failover

Chase provides its own Orbital system failover but for best performance and highest availability, Merchant should manage the failover process on their side.

- Chase exposes redundant hostname/port network endpoints to ensure high availability for the Chase Pay Service.
- Chase offers a Primary URL (will resolve to Chase's primary operational site) and Secondary URL (failover URL, which is always directed to resolve to the secondary site).
- To maximize availability, developers should include code to detect connectivity issues and, if necessary, HTTP errors, and, if necessary, temporarily switch to a failover URL.
- Communication with the primary hostname/port should be attempted periodically while in a state of failover.
- Recommended timeout interval for Primary URL is a max of 10 seconds. If no response is returned on the first attempt on the Primary URL, Merchants should failover to Secondary URL.

Orbital Authentication

Connection Username/Password authentication is required for all Chase Pay Service incoming requests.

- The connection Username and Password must be registered with Chase.
- The Username and Password must be passed in the message payload.
- The Username and Password must match the values registered with Chase in order to process transactions in the test and production environments.
- An HTTP 412 error is returned for all activity if the connection Username/Password is not registered with Chase.

NOTE: Orbital Gateway Merchants can leverage their existing Orbital authentication credentials

Credentials Hierarchy Options

Authentication credentials can exist at two different hierarchy levels depending on the Merchant's preference or need.

- At the Merchant level: One set of authentication credentials (such as a Connection User ID and Password) will be unique to one specific Merchant ID or TD ID.
- At the Chain or Company level: All Merchant IDs or unique locations will share one set of credentials for authentication under that specific chain or company.

If a Merchant expects to have more than one Merchant account using the Chase Pay Service:

- Set the Connection Username affiliated at the Chain/Company-level hierarchy within the service.
- This will ensure each time a new Merchant ID (MID) is added, the credentials will work as long as they are placed within the same Chain/Company.

If Connection Username is not placed within the same Chain/Company:

- The additional MIDs must be affiliated with the Connection Usernames.
- For example, all Stratus accounts (BIN 000001) are generally affiliated with their corresponding Company Number (formerly called MA #), resulting in all MIDs or Divisions under that Company being automatically affiliated.

NOTE: Merchants using an MPLS/VPN connection may not require connection Username/Password credentials to authenticate to the Chase Pay Service.

NOTE: IP authentication via Internet is available for existing Orbital Gateway Merchants who are already using this option.

Implementation Review

By now Merchant should have completed the following:

- *Electing to use SOAP or XML*
- *Understanding the options for the security requirements*
- *Having access to the WSDL or the XSD accordingly*
- *Connecting to the Chase Pay Service via the Internet or MPLS/VPN options*
- *Deciding on credential hierarchy options*
- *Creation and registration of Connection User ID and Password with Chase Merchant Services*
- *Handling of connection failovers*

Next Steps:

- *Start the Backend Wallet Integration*

Chase Pay Integration Points

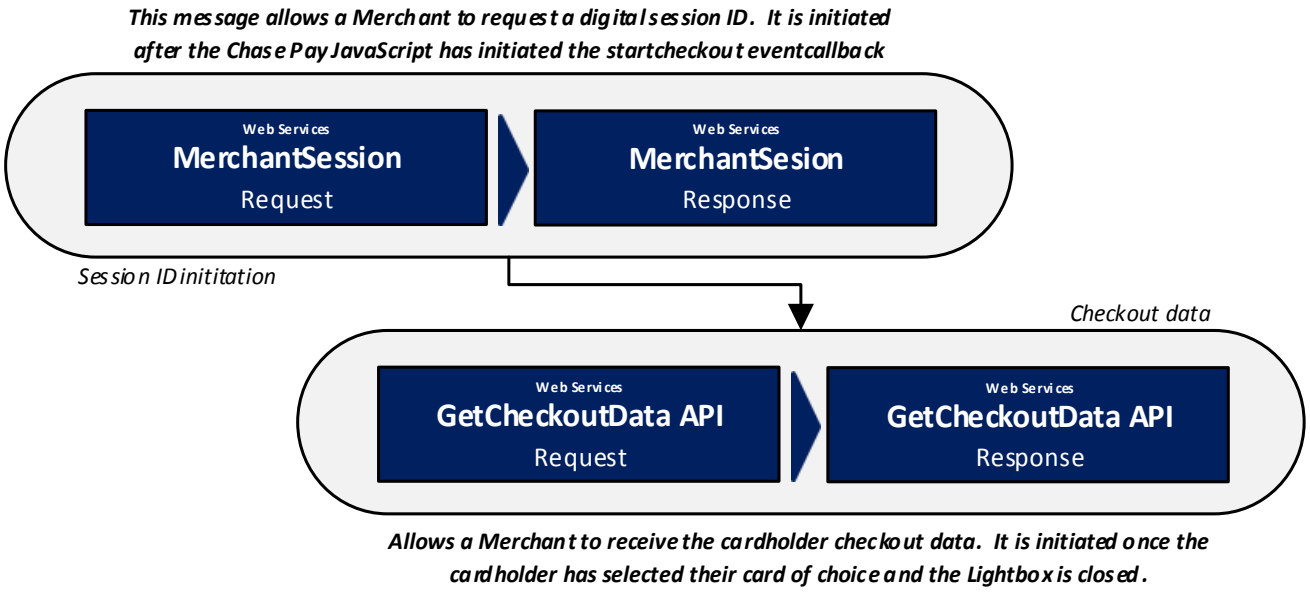
A FRONT END – Chase Pay button and Lightbox

B BACK END – Payment data retrieval API

C ACQUIRING – TXN processing

Chapter 4 API Calls and Payment Data Retrieval

Chase Pay Services API will communicate the payment data used by the Merchant to request a payment authorization. A summary of the two API's used is provided in the following diagram:



[REST OF THIS PAGE IS INTENTIONALLY BLANK]

SOAP Web Service Implementation

MerchantSession Data Message - SOAP

MerchantSession - SOAP Webservice - Request - Example

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns="urn:ws.paymentech.net/PaymentechGateway"
    xmlns:dw="urn:ws.paymentech.net/DigitalWalletService">
    <SOAP-ENV:Body>
        <MerchantSession xmlns="urn:ws.paymentech.net/DigitalWalletService">
            <MerchantSessionRequest>
                <orbitalConnectionUsername>XXXXXXX1</orbitalConnectionUsername>
                <orbitalConnectionPassword>YYYYYYy1</orbitalConnectionPassword>
                <version>1.6</version>
                <bin>000001</bin>
                <MerchantID>XXXXXX</MerchantID>
                <terminalID>001</terminalID>
                <merchRequestID>TX874665v-12345</merchRequestID>
                <dbaName>Joan's Cooking Show</dbaName>
                <paymentMethods>
                    <item>VI</item>
                    <item>CZ</item>
                    <item>CR</item>
                </paymentMethods>
                <consumerIP>111.111.111.111</consumerIP>
                <shipAddressInd>N</>
                <billAddressInd>Y</billAddressInd>
                <contactInfoInd>N</contactInfoInd>
                <shipToPostOfficeBoxInd>N</shipToPostOfficeBoxInd>
                <shipToMilitaryBaseInd>N</shipToMilitaryBaseInd>
      </MerchantSessionRequest>
        </MerchantSession>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

MerchantSession Request (Inbound from Merchant)

Element Name	Description	Data Type	Length	Required?
<orbitalConnectionUsername>	Connection Username set up for Merchant ID on Chase Merchant Services system Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Not case-sensitive Must match exactly what is registered on Chase Merchant Services system	Alpha Numeric	Variable Min 8 Max 32	M
<orbitalConnectionPassword>	Connection Password used in conjunction with Username Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Password is case-sensitive and Must match exactly what is registered on Chase Merchant Services system	Alpha Numeric	Variable Min 8 Max 32	M
<version>	Version of WSDL being used for the SOAP message If no version is specified, WSDL 1.0 will be used	Alpha Numeric	Variable Max 5	O
<bin>	Transaction Routing Definition Assigned by Chase Merchant Services Allowed values:	Numeric	6	M

Element Name	Description	Data Type	Length	Required?
	• Stratus = 000001 • Tandem = 000002			
<merchantID>	Merchant Account Number • If Stratus boarded Merchant, then this is a 6 digit Transaction Division Number • If Tandem boarded Merchant, then this is a 12 digit (max) PNS Merchant ID	Alpha Numeric	Variable Either 6 or 12	M
<terminalID>	Merchant Terminal ID Assigned by Chase Merchant Services Allowed values: • Stratus = 001 • Tandem = 001 to 999	Alpha	3	M
<mechRequestID>	Merchant request unique identifier defined by Merchant A new value should be generated for each request The valid characters include: • abcdefghijklmnopqrstuvwxyz • ABCDEFGHIJKLMNOPQRSTUVWXYZ • 0123456789 • _ - $ @ & and a space character, though the space character and _ cannot be the leading character	Alpha Numeric	Variable Max 22	M
<dbaName>	Merchant must populate this field with their consumer friendly name as it will be presented in the Chase Pay Lightbox when used at the Merchant's website	Alpha	Variable	O*

Element Name	Description	Data Type	Length	Required?
	**Although element is optional, Chase Pay implementation will not be certified for production unless this element is included in API*			
<paymentMethods>	Methods of Payments to be supported in digital wallet • MOPs on file will be used if left empty. • Invalid or unsupported MOPS will not be presented in the digital wallet. An error will be returned if no valid MOPs are provided. Submitted as a complex type: • VI for Visa • CR for ChaseNet Sig Debit • CZ for ChaseNet Credit These values affect the MOP's returned in the GetCheckOutResponse.	Alpha	2	O
<consumerIP>	IP address for consumer's device Standard format xxx.xxx.xxx.xxx Currently configured to support IPv4. Merchant should code to eventually support IPv6.	Alpha Numeric	Variable Max 20	M
<shipAddressInd>	Indicates whether to return shipping address in the getCheckoutDataResponse message Y = Yes **N = No** (Recommended if address is collected prior to reaching Chase Pay) When this element is set to N, the Lightbox does not provide cardholder the option to select shipping address at all.	Alpha	1	M

Element Name	Description	Data Type	Length	Required?
<billAddressInd>	Indicates whether to return billing address in the getCheckoutDataResponse message Allowed values **Y = Yes** (Recommended) N = No	Alpha Numeric	1	M
<contactInfoInd>	Indicates whether to return consumer contact information (consumer email address and phone number) in the getCheckoutDataResponse message Allowed values: Y = Yes **N = No** (Recommended if address is collected prior to reaching Chase Pay)	Alpha	1	M
<shipToPostOfficeBoxInd>	Indicates whether to return Post Office Box information in the getCheckoutDataResponse message Allowed values: Y = Yes **N = No** (Recommended)	Alpha	1	O
<shipToMilitaryBaseInd>	Indicates whether to return military base information in the getCheckoutDataResponse message Allowed values: Y = Yes **N = No** (Recommended)	Alpha	1	O

MerchantSession - SOAP Webservice - Response - Example


```xml
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns="urn:ws.paymentech.net/PaymentechGateway"
    xmlns:dw="urn:ws.paymentech.net/DigitalWalletService">
    <SOAP-ENV:Body>
        <MerchantSessionResponse>
            <return>
                <MerchantID>XXXXXX</MerchantID>
                <merchRequestID>TX874665v-12345</merchRequestID>
                <digitalSessionID>
DLgsxiFx6OHgLjoq9ijpVdnF8Kz6Bu2PAoCuFGw25HXu6TyqdeEpD2T8OT/6+YFtdtv34HqZG0Xo7X97KyOBow==</dig
italSessionID>
                <procStatus>0</procStatus>
                <procStatusMessage>Success</procStatusMessage>
            </return>
        </MerchantSessionResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


[REST OF THIS PAGE IS INTENTIONALLY BLANK]

MerchantSessionResponse (Outbound to Merchant)

Element name	Description	Data Type	Length	Required?
<version>	Version of WSDL being used for the SOAP message	Alpha Numeric	Variable Max 5	O
<merchantID>	Echo back from request	Alpha Numeric	Variable Either 6 or 12	M
<merchRequestID>	Echo back from request	Alpha Numeric	Variable Max 22	M
<digitalsessionID>	Value generated by Chase Pay Service Consists of a combination of security elements Required for consumer redirect	Alpha Numeric	Variable Min 200 Max 500	M
<procStatus>	Process Status Identifies whether requests were processed successfully: • 0 = Success • >0 = Error Refer to pages 77 and 78 in this document for details on codes and descriptions.	Numeric	Variable Up to 6	M
<procStatusMessage>	Descriptive text message associated with Process Status error	Alpha	Variable	M

NOTE: The <digitalSessionID> field will be passed back to the Chase Pay Event Callback JavaScript so that it can be validated against the open session.

GetCheckout Data Message - SOAP

GetCheckoutData - SOAP Webservice - Request - Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns="urn:ws.paymentech.net/PaymentechGateway"
    xmlns:dw="urn:ws.paymentech.net/DigitalWalletService">
    <SOAP-ENV:Body>
        <GetCheckoutData xmlns="urn:ws.paymentech.net/DigitalWalletService">
            <getCheckoutDataRequest>
                <orbitalConnectionUsername>XXXXXXX1</orbitalConnectionUsername>
                <orbitalConnectionPassword>YYYYYYy1</orbitalConnectionPassword>
                <bin>000001</bin>
                <MerchantID>XXXXXX</MerchantID>
                <terminalID>001</terminalID>
                <merchRequestID>TX874665v-12345</merchRequestID>
                <digitalSessionID>
DLgsxiFx6OHgLjoq9ijpVdnF8Kz6Bu2PAoCuFGw25HXu6TyqdeEpD2T8OT/6+YFtdtv34HqZG0Xo7X97KyOBow==</dig
italSessionID>
                <consumerIP>111.111.111.111</consumerIP>
            </getCheckoutDataRequest>
        </GetCheckoutData>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

GetCheckoutData Request (Inbound from Merchant)

Element Name	Description	Data Type	Length	Required?
<orbitalConnectionUsername>	Connection Username set up for Merchant ID on Chase Merchant Services system Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Not case-sensitive Must match exactly what is registered on Chase Merchant Services system	Alpha-numeric	Variable Min: 8 Max 32	M
<orbitalConnectionPassword>	Connection Password used in conjunction with Username Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Password is case-sensitive Must match exactly what is registered on Chase Merchant Services system	Alpha-numeric	Variable Min: 8 Max 32	M
<version>	Version of WSDL being used for the SOAP message If no version is specified, WSDL 1.0 will be used	Alpha Numeric	Variable Max = 5	O
<bin>	Transaction Routing Definition Assigned by Chase Merchant Services Allowed values:	Numeric	6	M

Element Name	Description	Data Type	Length	Required?
	• Stratus = 000001 • Tandem = 000002			
<merchantID>	Merchant Account Number • If Stratus boarded Merchant, then this is a 6 digit Transaction Division Number • If Tandem boarded Merchant, then this is a 12 digit (max) PNS Merchant ID	Alpha Numeric	Variable Either 6 or 12	M
<terminalID>	Merchant Terminal ID Assigned by Chase Merchant Services Allowed values: • Stratus = 001 • Tandem = 001 to 999	Numeric	3	M
<merchRequestID>	Merchant request unique identifier defined by Merchant A new value should be generated for each request The valid characters include: • abcdefghijklmnopqrstuvwxyz • ABCDEFGHIJKLMNOPQRSTUVWXYZ • 0123456789 • _ - $ @ & and a space character, though the space character and _ cannot be the leading character NOTE: Send this same value in the Merchant Order Number field of the authorization request.	Alpha Numeric	Variable Max 22	M

Element Name	Description	Data Type	Length	Required?
<digitalSessionID>	Session ID is provided Assumed that this will be the same digital session ID that was returned in MerchantSessionResponse.	Alpha Numeric	Variable Min 200 Max 500	M
<consumerIP>	IP address for consumer's device Standard format xxx.xxx.xxx.xxx Currently configured to support IPv4. Merchant should code to eventually support IPv6.	Alpha Numeric	Variable Max 20	M

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

GetCheckoutData - SOAP Webservice - Response - Example


```xml
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns="urn:ws.paymentech.net/PaymentechGateway"
    xmlns:dw="urn:ws.paymentech.net/DigitalWalletService">
    <SOAP-ENV:Body>
        <GetCheckoutDataResponse>
            <return>
                <MerchantID>XXXXXX</MerchantID>
                <merchRequestID>TX874665v-12345</merchRequestID>
                <accountNumber>4055000000001111</accountNumber>
                <expDate>0619</expDate>
                <paymentCryptogram>99999999999954545454545454</paymentCryptogram>
                <eciIndicator>7</eciIndicator>
                <tokenRequestorID>12345678901</tokenRequestorID>
                <billName>Jane Doe</billName>
                <billAddress1>123 Test Street</billAddress1>
                <billAddress2>Apt 512</billAddress2>
                <billCity>Tampa</billCity>
                <billState>FL</billState>
                <billZip>33607</billZip>
                <billCountry>USA</billCountry>
               <accountMask>************1234</accountMask>
                   <procStatus>0</procStatus>
                      <procStatusMessage>Success</procStatusMessage>
                         </return>
                     </GetCheckoutDataResponse>
                  </SOAP-ENV:Body>
              </SOAP-ENV:Envelope>
```


[REST OF THIS PAGE IS INTENTIONALLY BLANK]

GetCheckoutData Response – (Outbound to Merchant)

Element Name	Description	Data Type	Length	Required?
<version>	Version of WSDL being used for the SOAP message	Alpha Numeric	Variable Max 5	O
<merchantID>	Echo back from request	Alpha Numeric	Variable Either 6 or 12	M
<merchRequestID>	Echo back from request	Alpha Numeric	Variable Max 22	M
<accountNumber>	Cardholder account number in the form of a DPAN (brand Token) Required to submit an authorization request	Numeric	Variable Max = 19	M
<expDate>	DPAN expiration date Format: MMYY Required to submit an authorization request	Numeric	4	M
<paymentCryptogram>	The cryptogram associated with the Account Number Required to submit an authorization request Submitted in the respective VbV field in the authorization request	Alpha	Variable Max 40	M
<eciIndicator>	ECI indicator associated with the request Required to submit an authorization request Submitted in the respective ECI Indicator field in the authorization request	Numeric	Variable Max 2	M
<tokenRequestorID>	Token Requestor ID (TRI) which needs to be submitted in the authorization. Submitted in the respective Token Requestor ID field in the authorization request	Alpha Numeric	11	M

Element Name	Description	Data Type	Length	Required?
<shipName>	Consumer shipping name This field corresponds to the consumer's name for shipping purposes and is the full name in one field (First, Middle and Last Name).	Alpha	Variable Max 30	O
<shipFirstName>	Consumer shipping first name	Alpha	Variable Max 40	O
<shipMiddleName>	Consumer shipping middle name	Alpha	Variable Max 40	O
<shipLastName>	Consumer shipping last name	Alpha	Variable Max 40	O
<shipAddress1>	Consumer shipping address 1	Alpha Numeric	Variable Max 30	O
<shipAddress2>	Consumer shipping address 2	Alpha Numeric	Variable Max 30	O
<shipCity>	Consumer shipping city	Alpha	Variable Max 20	O
<shipState>	Consumer shipping state	Alpha	2	O
<shipZip>	Consumer shipping ZIP code ZIP or extended ZIP+4	Alpha Numeric	Variable Max 10	O
<shipCountry>	Consumer shipping country code 3 character country code Country code to where the goods are being shipped.	Alpha	3	O
<billName>	Consumer billing name	Alpha	Variable Max 30	O
<billFirstName>	Consumer billing first name	Alpha	Variable Max 40	O
<billMiddleName>	Consumer billing middle name	Alpha	Variable Max 40	O

Element Name	Description	Data Type	Length	Required?
<billLastName>	Consumer billing last name	Alpha	Variable Max 40	O
<billAddress1>	Consumer billing address 1	Alpha Numeric	Variable Max 30	O
<billAddress2>	Consumer billing address 2	Alpha Numeric	Variable Max 30	O
<billCity>	Consumer billing city	Alpha	Variable Max 20	O
<billState>	Consumer billing state	Alpha	2	O
<billZip>	Consumer billing ZIP code ZIP or extended ZIP+4	Alpha Numeric	Variable Max 10	O
<billCountry>	Consumer billing country code 3 character country code	Alpha	3	O
<consumerEmail>	Consumer Email	Alpha Numeric	Variable Max 50	O
<consumerPhone>	Consumer Phone International phone numbers begin with (+)	Alpha Numeric	20	O
<accountMask>	Masked account number with the last 4 digits of the clear FPAN. Needed for receipt. Example: **********1234 **NOTE:** The asterisks and the last four digits are returned.	Alpha Numeric	Variable Max 19	M
<procStatus>	Process Status Identifies whether requests were process successfully: • 0 Success • >0 Error – See Response Handling chapter for list of responses	Numeric	Variable Up to 6	M

Element Name	Description	Data Type	Length	Required?
	Refer to pages 77 and 78 in this document for details on codes and descriptions.			
<procStatusMessage>	Descriptive text message associated with Process Status error	Alpha	Variable	M

SOAP Fault Element

- An error message from a SOAP message can be carried inside a <Fault> element.
- If a <Fault> element is present, it will appear as a child element of the <Body> element.
- A <Fault> element can only appear once in a SOAP message.

Sub elements of the SOAP Fault element

- <faultcode> - For errors generated by the Chase Pay Service, this value is SOAP-ENV:Server.
- <faultstring> - For Chase Pay Service-generated errors (where the Faultcode = Server), this is a concatenation of the Service Error Code and Description.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns="urn:ws.paymentech.net/DigitalWalletService">
    <SOAP-ENV:Body id="_0">
        <SOAP-ENV:Fault>
            <faultcode> SOAP-ENV:Server </faultcode>
            <faultstring> 46514 Error. Digital Session ID Error </faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

XML Implementation

MerchantSession Data Message – XML

MerchantSession - XML - Request - Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request>
  <MerchantSession>
    <OrbitalConnectionUsername>XXXXXXX2</OrbitalConnectionUsername>
    <OrbitalConnectionPassword>YYYYYYy2</OrbitalConnectionPassword>
    <BIN>000001</BIN>
    <MerchantID>XXXXXX</MerchantID>
    <TerminalID>001</TerminalID>
    <MerchRequestID>TX874665v-12345</MerchRequestID>
    <DBAName>testDBA</DBAName>
    <PaymentMethods>
      <item>VI</item>
      <item>CZ</item>
      <item>CR</item>
    </PaymentMethods>
    <ConsumerIP>111.111.111.111</ConsumerIP>
    <ShipAddressInd>N</ShipAddressInd>
    <BillAddressInd>Y</BillAddressInd>
    <ContactInfoInd>N</ContactInfoInd>
  </MerchantSession>
</Request>
```

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

MerchantSessionRequest (Inbound from Merchant)

Element Name	Description	Data Type	Length	Required?
Request	Required XML Parent Tag	N/A	N/A	M
MerchantSession	XML tag that defines the transaction as a Merchant Session request	N/A	N/A	M
<OrbitalConnectionUsername>	Connection Username set up for Merchant ID on Chase Merchant Services system Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Not case-sensitive Must match exactly what is registered on Chase Merchant Services system	Alpha Numeric	Variable Min 8 Max 32	M
<OrbitalConnectionPassword>	Connection Password used in conjunction with Username Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Password is case-sensitive and Must match exactly what is registered on Chase Merchant Services system	Alpha Numeric	Variable Min 8 Max 32	M
<BIN>	Transaction Routing Definition Assigned by Chase Merchant Services Allowed values:	Numeric	6	M

Element Name	Description	Data Type	Length	Required?
	• Stratus = 000001 • Tandem = 000002			
<MerchantID>	Merchant Account Number • If Stratus boarded Merchant, then this is a 6 digit Transaction Division Number • If Tandem boarded Merchant, then this is a 12 digit (max) PNS Merchant ID	Alpha Numeric	Variable Either 6 or 12	M
<TerminalID>	Merchant Terminal ID Assigned by Chase Merchant Services Allowed values: • Stratus = 001 • Tandem = 001 to 999	Numeric	3	M
<MerchRequestID>	Merchant request unique identifier defined by Merchant A new value should be generated for each request The valid characters include: • abcdefghijklmnopqrstuvwxyz • ABCDEFGHIJKLMNOPQRSTUVWXYZ • 0123456789_ • - $ @ & and a space character, though the space character and _ cannot be the leading character NOTE: Send this same value in the Merchant Order Number field of the authorization request	Alpha Numeric	Variable Max 22	M
<DBAName>	Merchant must populate this field with their consumer friendly name as it will be	Alpha	Variable	O*

Element Name	Description	Data Type	Length	Required?
	presented in the Chase Pay Lightbox when used at the Merchant's website *Although element is optional, Chase Pay implementation will not be certified for production unless this element is included in API			
<PaymentMethods>	Methods of Payment to be supported in digital wallet MOP's on file will be used if empty. • Invalid or unsupported MOPS will not be presented in the digital wallet. An error will be returned if no valid MOP's are provided. Submitted as complex type. • VI for VISA • CR for ChaseNet Sig Debit • CZ for ChaseNet Credit These values affect the MOP's returned to the GetCheckOut response.	Alpha	2	O
<ConsumerIP>	IP address for consumer's device Standard format xxx.xxx.xxx.xxx Currently configured to support IPv4. Merchant should code to eventually support IPv6.	Alpha Numeric	Variable Max 20	M
<ShipAddressInd>	Indicates whether to return shipping address in the getCheckoutData response message	Alpha	1	M

Element Name	Description	Data Type	Length	Required?
	Y = Yes **N = No** (Recommended if address is collected prior to reaching Chase Pay) When this element is set to N, the Lightbox does not provide cardholder the option to select shipping address at all.			
<BillAddressInd>	Indicates whether to return billing address in the getCheckoutData response message Allowed values **Y = Yes** (Recommended) N = No	Alpha	1	M
<ContactInfoInd>	Indicates whether to return consumer contact information (consumer email address and phone number) in the getCheckoutData response message Allowed values: Y = Yes **N = No** (Recommended if address is collected prior to reaching Chase Pay)	Alpha	1	M
<ShipToPostOfficeBoxInd>	Indicates whether to return Post Office Box information in the getCheckoutData response message Allowed values: Y = Yes **N = No** (Recommended)	Alpha	1	O

Element Name	Description	Data Type	Length	Required?
<ShipToMilitaryBaseInd>	Indicates whether to return military base information in the getCheckoutData response message Allowed values: Y = Yes **N = No** (Recommended)	Alpha	1	O

NOTE: Optional fields that are sent with a null value will default to "**N**".

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

MerchantSession - XML - Response - Example


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <MerchantSessionResp>
    <MerchantID>XXXXXX</MerchantID>
    <MerchRequestID>TX874665v-12345</MerchRequestID>
    <DigitalSessionID>P6Uu/y/Ct4OTDgI9dcIEaEcMx2L7p1u42bhatWtQJmGLvnBj1alNSbDDKwNXS
+VZLzmSOgJYoK1Ah8jdPLxIyHM8OYYK8fT/VkGHq3y6ZhE=</DigitalSessionID>
    <ProcStatus>0</ProcStatus>
    <StatusMsg>Success</StatusMsg>
  </MerchantSession>
</Response>
```


[REST OF THIS PAGE IS INTENTIONALLY BLANK]

MerchantSession Response (Outbound to Merchant)

Element Name	Description	Data Type	Length	Required?
Response	Required XML Parent Tag	N/A	N/A	M
MerchantSession	XML tag that defines the transaction as a Merchant Session response	N/A	N/A	M
<MerchantID>	Echo back from request	Alpha Numeric	Variable Either 6 or 12	M
<MerchRequestID>	Echo back from request	Alpha Numeric	Variable Max 22	M
<DigitalSessionID>	Value generated by Chase Pay Service Consists of a combination of security elements Required for consumer redirect	Alpha Numeric	Variable Min 200 Max 500	M
<ProcStatus>	Process Status Identifies whether requests were processed successfully: • 0 = Success • >0 = Error Refer to pages 77 and 78 in this document for details on codes and descriptions.	Numeric	Variable Max 6	M
<StatusMsg>	Descriptive text message associated with Process Status error	Alpha	Variable	M

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

GetCheckout Data Message - XML

GetCheckoutData - XML - Request - Example


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request>
  <GetCheckoutDataRequest>
    <OrbitalConnectionUsername>XXXXXXX2</OrbitalConnectionUsername>
    <OrbitalConnectionPassword>YYYYYYy2</OrbitalConnectionPassword>
    <BIN>000001</BIN>
    <MerchantID>XXXXXXX</MerchantID>
    <TerminalID>001</TerminalID>
    <MerchRequestID>TX874665v-12345</MerchRequestID>
    <DigitalSessionID>P6Uu/y/Ct4OTDgI9dcIEaEcMx2L7p1u42bhatWtQJmGLvnBj1alNSbDDKwNXS+VZLzmSOgJ
YoK1Ah8jdPLxIyHM8OYYK8fT/VkGHq3y6ZhE=</DigitalSessionID>
    <ConsumerIP>111.111.111.111</ConsumerIP>
  </GetCheckoutDataRequest>
</Request>
```


[REST OF THIS PAGE IS INTENTIONALLY BLANK]

GetCheckoutData - XML - Request - Description

Element Name	Description	Data Type	Length	Required?
Request	Required XML Parent Tag	N/A	N/A	M
GetCheckoutData	XML tag that defines the transaction as a GetCheckoutData request	N/A	N/A	M
<OrbitalConnectionUsername>	Connection Username set up for Merchant ID on Chase Merchant Services system Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Not case-sensitive Must match exactly what is registered on Chase Merchant Services system	Alpha Numeric	Variable Min 8 Max 32	M
<OrbitalConnectionPassword>	Connection Password used in conjunction with Username Formats: • Between 8–32 characters (a-z, A-Z, 0-9) • Minimum 1 number • No leading, trailing, or embedded spaces • Password is case-sensitive Must match exactly what is registered on Chase Merchant Services system	Alpha Numeric	Variable Min 8 Max 32	M
<BIN>	Transaction Routing Definition Assigned by Chase Merchant Services Allowed values:	Numeric	6	M

Element Name	Description	Data Type	Length	Required?
	• Stratus = 000001 • Tandem = 000002			
<MerchantID>	Merchant Account Number • If Stratus boarded Merchant, then this is a 6 digit Transaction Division Number • If Tandem boarded Merchant, then this is a 12 digit (max) PNS Merchant ID	Alpha Numeric	Variable Either 6 or 12	M
<TerminalID>	Merchant Terminal ID Assigned by Chase Merchant Services Allowed values: • Stratus = 001 • Tandem = 001 to 999	Numeric	3	M
<MerchRequestID>	Merchant request unique identifier defined by Merchant A new value should be generated for each request The valid characters include: • abcdefghijklmnopqrstuvwxyz • ABCDEFGHIJKLMNOPQRSTUVWXYZ • 0123456789 • _ - $ @ & and a space character, though the space character and _ cannot be the leading character NOTE: Send this same value in the Merchant Order Number field of the authorization request	Alpha Numeric	Variable Max 22	M
<DigitalSessionID>	Session ID is provided	Alpha Numeric	Variable	M

Element Name	Description	Data Type	Length	Required?
	Assumed that this will be the same digital session ID that was returned in MerchantSession response		Min 200 Max 500	
<ConsumerIP>	IP address for consumer's device Standard format xxx.xxx.xxx.xxx Currently configured to support IPv4. Merchant should code to eventually support IPv6.	Alpha Numeric	Variable Max 20	M

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

GetCheckoutData - XML - Response - Example


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <GetCheckoutDataResponse>
    <MerchantID>XXXXXX</MerchantID>
    <MerchRequestID>TX874665v-12345</MerchRequestID>
    <AccountNum>4055000000001111</AccountNum>
    <ExpDate>0619</ExpDate>
    <PaymentCryptogram>99999999999954545454545454</PaymentCryptogram>
    <EciIndicator>7</EciIndicator>
    <TokenRequestorID>12345678901</TokenRequestorID>
    <ShipName></ShipName>
    <ShipAddress1></ShipAddress1>
    <ShipAddress2></ShipAddress2>
    <ShipCity></ShipCity>
    <ShipState></ShipState>
    <ShipZip></ShipZip>
    <ShipCountry></ShipCountry>
    <BillName>Jane Doe</BillName>
    <BillAddress1>4900 MEMORIAL HWY</BillAddress1>
    <BillAddress2></BillAddress2>
    <BillCity>TAMPA</BillCity>
    <BillState>FL</BillState>
    <BillZip>33634</BillZip>
    <BillCountry>USA</BillCountry>
    <ConsumerEmail>XYZ@email.com</ConsumerEmail>
    <ConsumerPhone>XXXXXXXX</ConsumerPhone>
    <ProcStatus>0</ProcStatus>
    <StatusMsg>Success</StatusMsg>
    <AccountMask>************1234</AccountMask>
  </GetCheckoutDataResponse>
</Response>
```


[REST OF THIS PAGE IS INTENTIONALLY BLANK]

GetCheckoutData - XML - Response – Description

Element Name	Description	Data Type	Length	Required?
Response	Required XML Parent Tag	N/A	N/A	M
GetCheckoutData	XML tag that defines the transaction as a GetCheckoutData response	N/A	N/A	M
<MerchantID>	Echo back from request	Alpha Numeric	Variable Either 6 or 12	M
<MerchRequestID>	Echo back from request	Alpha Numeric	Variable Max 22	M
<AccountNum>	Cardholder account number in the form of a DPAN (brand Token) Required to submit an authorization request	Numeric	Variable Max 19	M
<ExpDate>	DPAN expiration date Format: MMYY Required to submit an authorization request	Numeric	4	M
<PaymentCryptogram>	The cryptogram associated with the Account Number Required to submit an authorization request Submitted in the respective VbV field in the authorization request	Alpha	Variable Max 40	M
<EciIndicator>	ECI indicator associated with the request Required to submit an authorization request Submitted in the respective ECI Indicator field in the authorization request	Numeric	Variable Max 2	M
<TokenRequestorID>	Required to submit an authorization request	Alpha	11	M

Element Name	Description	Data Type	Length	Required?
	Submitted in the respective Token Requestor ID field in the authorization request			
<ShipName>	Consumer shipping name This field corresponds to the consumer's name for shipping purposes and is the full name in one field (First, Middle and Last Name).	Alpha	Variable Max 30	O
<ShipFirstName>	Consumer shipping first name	Alpha	Variable Max 40	O
<ShipMiddleName>	Consumer shipping middle name	Alpha	Variable Max 40	O
<ShipLastName>	Consumer shipping last name	Alpha	Variable Max 40	O
<ShipAddress1>	Consumer shipping address 1	Alpha Numeric	Variable Max 30	O
<ShipAddress2>	Consumer shipping address 2	Alpha Numeric	Variable Max 30	O
<ShipCity>	Consumer shipping city	Alpha	Variable Max 20	O
<ShipState>	Consumer shipping state	Alpha	2	O
<ShipZip>	Consumer shipping ZIP code ZIP or extended ZIP+4	Alpha Numeric	Variable Max 10	O
<ShipCountry>	Consumer shipping country code 3 character country code Country code to where the goods are being shipped.	Alpha	3	O
<BillName>	Consumer billing name	Alpha	Variable Max 30	O

Element Name	Description	Data Type	Length	Required?
<BillFirstName>	Consumer billing first name	Alpha	Variable Max 40	O
<BillMiddleName>	Consumer billing middle name	Alpha	Variable Max 40	O
<BillLastName>	Consumer billing last name	Alpha	Variable Max 40	O
<BillAddress1>	Consumer billing address 1	Alpha Numeric	Variable Max 30	O
<BillAddress2>	Consumer billing address 2	Alpha Numeric	Variable Max 30	O
<BillCity>	Consumer billing city	Alpha	Variable Max 20	O
<BillState>	Consumer billing state	Alpha	2	O
<BillZip>	Consumer billing ZIP code ZIP or extended ZIP+4	Alpha Numeric	Variable Max 10	O
<BillCountry>	Consumer billing country code 3 character country code	Alpha	3	O
<ConsumerEmail>	Consumer Email	Alpha	Variable Max 50	O
<ConsumerPhone>	Consumer Phone International phone numbers begin with (+)	Alpha Numeric	20	O
<AccountMask>	Masked account number with the last 4 digits of the clear FPAN. Needed for receipt. Example: **********1234 **NOTE:** The asterisks and the last four digits are returned.	Alpha Numeric	Variable Max 19	
<ProcStatus>	Process Status Identifies whether requests were process successfully:	Numeric	Variable Up to 6	M

Element Name	Description	Data Type	Length	Required?
	• 0 Success • >0 Error – See Response Handling chapter for list of responses Refer to pages 77 and 78 in this document for details on codes and descriptions.			
<StatusMsg>	Descriptive text message associated with Process Status error	Alpha	Variable	M

Error Handling

The table below lists the error codes <procStatus> or <ProcStatus> and descriptions <procStatusMessage> or <StatusMsg> contained within the API responses:

Error Code	Description
0	Success
9716	Security Information is Missing
9717	Security Information – agent/chain/Merchant is missing
20400	Invalid Request
20403	Forbidden: SSL Connection Required
20408	Request Timed Out
20412	Precondition Failed: Security Information is Missing
20500	Internal Server Error
20502	Connection Error
20503	Server Unavailable: Please Try Again Later
40001	The field is missing, invalid, or has exceeded the max length.
40002	Please validate input and try again.
40004	Not configured for Chase Pay. Please contact your Relationship Manager.
40006	Error – Please try again.
40007	Merchant Not Authorized To Perform the Transaction.
42000	Downstream System Error.
42012	Proxy error - please try again later.
42035	We’re unable to process your request because we’re currently performing system maintenance. Please try again later.
44001	System error.
44032	Device Signature Bad Format.
44089	Dynamic Token Mismatch is not within tolerance.
44093	Max number of tokens per application has been exceeded.
44094	Not configured for Chase Pay. Please contact your Relationship Manager.
46501	Not configured for Chase Pay. Please contact your Relationship Manager.
46503	Please validate either session id or transaction reference key and resubmit.
46510	Downstream System Error.

Error Code	Description
46511	Digital session id is either missing or invalid. Please try again.
46512	Invalid IP.
46513	Your session has timed out, please try again.
46514	Digital session ID is either missing or invalid. Please try again.
46521	Merchant customer id does not match our records.
46576	Invalid transaction authorization code.
46577	Invalid Visa transaction ID.
46578	Invalid payment confirmation status.
46582	Please validate either session id or txn ref key and resubmit.
46588	Transaction is not active.
46593	Not configured for Chase Pay. Please contact your Relationship Manager.
49999	Unknown error.

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

Putting it All Together – From Using the Chase Pay Button to Transaction Mapping

Step 1: *Consumer clicks on Chase Pay button to initiate payment and checkout*

Step 2: *A Digital Session ID is created via the* ***MerchantSession*** *API and the Digital Session ID is validated*

Step 3: *After Session ID validation, the Lightbox is displayed and the consumer uses their Chase.com credentials to authenticate to their Chase Pay digital wallet*

Step 4: *Consumer selects a payment option from their Chase Pay digital wallet*

Step 5: *Upon selection of payment option, Lightbox closes and a* ***GetCheckoutData request*** *is generated. The Digital Session ID is used to retrieve the consumer's selected payment details data.*

Step 6: *Upon validating the Digital Session ID sent via the request, the elements and fields below are returned to the Merchant's web server via* ***GetCheckoutData response*** *and mapped within the tokenized authorization request to ChaseNet [NEXT PAGE]:*

[REST OF THIS PAGE IS INTENTIONALLY BLANK]

Chase Pay E-commerce message Mapping Field

GetCheckOutData Response Field	Tandem ISO	Tandem UTF	Stratis Online	Stratus Batch	Orbital XML	Orbital SOAP
<PaymentCryptogram>	*Bit 48 - Subtag 48*	*Token VA Authentication Data Field*	*Inbound format indicator VA CAVV Field*	*Extension record EV1002, ECR002, or ECZ002 CAVV Field*	*<CAVV>*	*<verifyByVisaCAVV>*
<ECI>	*Bit 48 - Subtag E1*	*E-commerce indicator*	*Online detail record: transaction type field*	*"S" Record Input Transaction Type Field*	*<AuthenticationECIInd>*	*<authenticationECIInd>*
<TokenRequestorId> (TRI)	*Bit 48 - Subtag TK*	*Token "TK"*	*Inbound format indicator DN – Token Requestor ID field*	*Product Record PDN001 – Token Requestor ID Field*	*<TokenRequestorID>*	*<tokenRequestorID>*
<AccountNumber> (DPAN)	*Bit 2*	*Account number field*	*Account number field*	*"S" record Input Account Number field*	*<AccountNum>*	*<ccAccountNum>*

Chapter 5 Digital Account Management in the Chase Pay Wallet

As part of the Chase Pay service, Chase automatically updates any card account data within the Chase Pay wallet when data changes are made due to expirations, re-issuances, or lost/stolen reasons.

- The account information provided by Chase Pay is the most current and accurate information that Chase has on file for the given account.
- To ensure successful authorization, do not change or allow the editing of the payment information provided by Chase Pay.
- To benefit from the most current account payment information that Chase has, do not plan on storing the payment information details received from Chase Pay to be used in future non-Chase Pay transactions. The only exception to this is the processing of split payments.

NOTE: Chase Pay returns consumer cardholder personal identifiable information (PII) and it is the Merchant's responsibility to protect this data. Consumer PII can only be used for purposes of completing the transaction. The information provided by Chase Pay should not be used for any other purposes without first obtaining authorization from the consumer.

Chapter 6 Handling Chase Pay Transactions

This chapter provides an overview of how the Chase Pay solution works in different transaction scenarios.

Purchase Transactions

Merchant should handle Chase Pay purchase transactions as business as usual following transaction processing specifications for wallet tokenized transactions.

Purchase Cancellation

Merchant should handle Chase Pay void and cancellation transactions as business as usual following transaction processing specifications for wallet tokenized transactions.

Returns and Refunds

Merchant should handle returns and refunds for Chase Pay transactions as business as usual per Merchant's return policy.

Split Tender

Split tender refers to using more than one method of payment to process the transaction. Merchants should handle split tender transactions business as usual per Merchant's standard processes.

Split Shipments

Split Shipments or Ship Partial allows the Merchant to do a future fulfillment or to partial ship an order and settle only the portion that has been shipped.

- Bankcard regulations prohibit charging a consumer for goods prior to shipment.
- If a consumer purchases several items and multiple shipments occur on different days (back orders, etc.), then the order must be charged to the consumer in multiple increments.
- The ship partial transaction accommodates this situation.
- In the Split Shipment or Ship Partial scenario:
    - Merchants make an authorization service call for the total purchase amount, using:
        - <accountNumber>, the tokenized account number
        - <paymentCryptogram>, the original cryptogram
        - <eciIndicator>, ECI value as returned from the Chase Pay Service
    - Subsequent clearing transactions shall use:
        - <accountNumber>, the tokenized account number
        - <paymentCryptogram>, the original cryptogram
        - <eciIndicator>, ECI value as returned from the Chase Pay Service.

NOTE: Split shipment transaction treatment will be updated in future releases to support the Visa defined Merchant Initiated Transactions (MIT).

Recurring Payments

A recurring payment scenario occurs when Merchant obtains the consumer's explicit approval to be charged on the same payment card on a recurring basis.

- Typical examples include utility and subscriptions services.
- In the recurring payment scenario:
  - The first transaction looks like a regular single transaction, using ECI=7 (regular ecommerce https transaction) and the unique cryptogram as returned from the Chase Pay Service.
  - Subsequent transactions will use ECI=2 for recurring transactions.
  - Acquirer may need to do some additional configuration to convert the ECI codes into values that are recognized by their processing infrastructure.
- With brand (EMVCo) tokenization, recurring transactions work uninterrupted even in cases where a new card is issued (user activation required for debit cards).

NOTE: Recurring transactions treatment will be updated in future releases to support the Visa defined Merchant Initiated Transactions (MIT).

Disputes & Chargebacks

All Chase Pay enabled transactions are uniquely identified by Chase through ChaseNet and therefore offer zero consumer fraud. Disputed transactions that are not fraudulent follow standard chargeback resolution processes.

Handling Returns

Business as usual procedures should be followed with returns.

- The Merchant should use the Chase Pay provided information, including the tokenized account number, <accountNumber>, just as they process returned transactions.

- Use <accountMask> field to display to the consumer the last 4 digits of the account as printed on their plastic card.

Implementation Review

By now Merchant should have completed the following:

- Retrieval of the Chase Pay payment details using the APIs in the previous module
- Formulation your authorization processing requests to include the Chase Pay payment details
- Special transaction handling
- Cardholder PI data handling

Next Steps:

Merchant is now ready for certification.

Prior to commencing the certification process:

- Merchant needs to use the proper credentials that have been assigned for the certification environment
- Merchant needs to use the correct URL or address to connect to the Chase Pay Service certification environment
- Merchant needs to use the latest version of the WSDL

Chapter 7 Certification

Now that the implementation and integration is complete, the implementation needs to be certified prior to launching code into production.

- Check with the Chase Technical Implementations Manager to schedule testing and certification as necessary.
- The WSDL file from the web service interface only references the Primary URL from the certification environment. When moving from certification to production, Merchant must change the address manually.
- While the certification system is available for testing at all hours, it is only monitored for availability during business hours (8:00am EST – 5:30pm EST Monday – Friday).

NOTE: The certification environment has been designed to test and certify the functionality and it is not intended to be used for load testing.

Chapter 8 Other Resources

In addition to this specification, the following resources are available to complete any integration of Chase Pay.

Resource	Use	How to Obtain
Core acquiring spec	Needed if upgrading authorization format.	See developer center https://www.chasepaymentech.com/developercenter/
Test environment connectivity	Depending on a Merchant's existing setup the following may need to be setup: • Orbital credentials may need to be established. • Acquiring test environment connectivity.	Consult your Chase Technical Implementations Manager.
Self-test documents: • CPS Test Conditions (Tandem Platform) • Triggered Response (Stratus Platform) • Orbital Triggered Responses document for OGW to Stratus processing • Chase Pay test cases	These documents allow Merchants to trigger a wide variety of authorization responses in the Merchant Services certification environment. These cases can be used in optional testing that Merchants may wish to do outside of the official certification script.	Consult your Chase Technical Implementations Manager.

[END OF DOCUMENT]