CyberSource Cartridge

Version 1.2.0



04/01/2013

Contents

Summary 5

Component Overview 7

Functional Overview 7

Credit Card Authorization Service 7

Taxes 8

Address Verification Service (AVS) 9

Delivery Address Verification Service (DAV) 9

Bill Me Later (BML) 9

Decision Manager 10

Payment Tokenization 12

Payer Authentication 12

Full Authorization Reversal 13

Use Cases Scenarios 13

Credit Card Authorization 13

Taxes 16

Address Validation Service (AVS) 16

Delivery Address Verification Service (DAV) 17

BML 17

Decision Manager 18

Payment Tokenization 18

Payer Authorization 19

V.me by Visa 20

Limitations, Constraints 23

Compatibility 23

Implementation Guide 23

Custom Code 23

Credit Card Auth 23

Bill Me Later 25

Tax Service 26

Delivery Address Validation Service 28

Payment Tokenization Service 28

Full Authorization Reversal 28

Payer Authentication Service 29

V.me by Visa 30

Clickjacking Prevention 41

Order Export XML – New Fields 42

Site Configuration 44

Import Meta Data 44

Configure Site Preferences 45

Configure Site Preferences for V.me 49

Applying CyberSource Cartridge to the Site 50

Applying V.me Cartridge to the Site 50

Testing 51

Authorize Credit Card 51

Tax Service 52

Address Verification Service (AVS) 52

Delivery Address Verification Service (DAV) 52

Payment Tokenization 52

Full Authorization reversal 53

Device Fingerprint 53

Payer Authentication 53

V.me Authorization 53

V.me Get Checkout Details 53

V.me Get Transaction Details 53

V.me Confirm Purchase 53

V.me Authorization Reversal 54

V.me Capture 54

V.me Refund 54

V.me Initiate Request 54

Cartridges Structure and Reference 55

Pipelines 55

Scripts 55

Templates 57

Configuration Files 57

V.me Cartridge Structure 57

Typical Project Plan 59

Roles, Responsibilities 59

Typical Efforts and Timelines 59

Pre-Production Steps 61

CyberSource Site Preferences 62

V.me Site Preferences 64

Device Fingerprint 65

How does it work? 65

Setup: 65

Hints for the CsDeviceFingerprintRedirectionType: 66

Modified Scripts and pipelines for the device fingerprint 67

Known Issues 68

Supported Versions 68

CyberSource document links 69

Release History 70

# Summary

This document provides technical overview and implementation details for each CyberSource service integrated within Demandware platform. The CyberSource cartridge extends the functionality of Demandware Storefront, enabling real time access to CyberSource eCommerce transaction services listed below.

Credit Card Authorization – The CyberSource Credit Card Authorization service provides merchant with a mechanism to get authorization for the order amount. The authorization service validates the card based and authorize card for the order amount. For additional spam/fraud detection by CyberSource, a ‘device fingerprint’ could be submitted additionally, if configured.

CyberSource Address Verification (AVS) – The CyberSource AVS service provides merchants with a mechanism to reduce merchant banking fees, by verifying billing information before authorizing payment for customer purchases. Although the AVS service is automatically called during the authorization process, the behavior resulting from specific AVS responses and its interaction with the payment process is customizable through storefront configuration.

Tax Service – The CyberSource tax calculation service provides merchants with a complete tax calculation service according to and pursuant to domestic and international tax regulations, including but not limited to, district, city, county and state levels of governing tax authority.

Delivery Address Verification (DAV) Service – The CyberSource DAV service provides merchants with an optional mechanism to prevent, limit or correct faulty shipping information, related to improperly entered or formatted information from the customer. This service helps reduce the potential additional costs resulting from undeliverable or returned merchandise.

Bill Me Later Service – The Bill Me Later service from Bill Me Later, Inc., and available via your single connection to CyberSource, allows your customers to make purchases using an instant line of credit.

Decision Manager – CyberSource Decision Manager Service provides set of tools that merchant to evaluate rules and chose tool and return a decision of “Accept”, “Reject”, or “Review”. Merchant can also setup process to ignore certain rules when necessary.

Payment Tokenization – CyberSource Payment Tokenization Service provides set of tools to store customer and payment related sensitive data on secured cybersource hosted servers.

Payer Authentication – CyberSource Payer Authentication services enable you to add support to your web store for card authentication services, including Visa Verified by VisaSM, MasterCard® andMaestro® SecureCode™ (UK Domestic and international), American Express SafeKeySM.

V.me by Visa – V.me service by Visa is a service that lets customers easily checkout without entering their payment information. V.me simplifies the payment experience for customers. Merchants can add the V.me checkout button to their site, so customers can check out with a user name and password. Customers store their major credit or debit cards (Visa, MasterCard, American Express, Discover) and shipping addresses in their V.me account. To pay, customers click the V.me checkout button and sign into their account. They confirm their payment details and complete checkout in just a few clicks without leaving merchant’s site.

# Component Overview

## Functional Overview

### Credit Card Authorization Service

The credit card authorization service pipeline allows storefront application to request for credit authorization for the total order amount. The pipeline makes the credit card authorization web service call to CyberSource authorization service and receive confirmation about the availability of the funds.

The Demandware Cybersource–AuthorizeCreditCard pipeline populates the authorization request with ship-to, bill-to, credit card data, and purchase total data from the basket and invokes the authorization web service call using CyberSource web service API.

Credit Card Authorization sequence flow:

1. Creates CyberSource authorization request using ship-to, bill-to, credit card data, and purchase total data from the current basket.
2. If authorize Payer is configured, then make the authorize payer request, if not ignore and continue with the authorization request.
3. Create credit card authorization request.
4. If DAV is enabled, then set up DAV business rules, as needed.
5. Set up AVS Ignore Result business rule for request with AVS Ignore Flags specification, as needed.
6. Make actual service call to CyberSource Simple Order API.
7. If Delivery Address Verification is enabled, then:
   1. Capture pertinent DAV result information & DAV Reason Code
   2. If DAV fails and DAV On Failure is set to ‘REJECT’, then exit immediately with rejection response
8. If DAV On Failure is set to ‘APPROVE’ and the DAV Reason Code is a fail code (not 100), then:
   1. Exit immediately with declined or review response, as merchant defines
9. Capture pertinent AVS information
10. Validate authorization reason code and set corresponding end node, based on auth response code.

The list of activities depicted in the following diagram take place when API request is made for an online credit card authorization: [**Source, CyberSource Credit Card Service, October 2009**]

**Figure 1** Processing an Online Authorization [**Source, CyberSource Credit Card Service, October 2009**]



1. The customer places an order and provides the credit card number, the card expiration date, and other information about the card.

2   You send a request for authorization over a secure Internet connection. If the customer buys a digitally delivered product or service, you can request both the authorization and the capture at the same time. If the customer buys a physically fulfilled product, do not request the capture until you ship the product.

3   CyberSource validates the order information, then contacts your payment processor and requests authorization.

4   The processor sends the transaction to the card association, which routes it to the issuing bank for the customer’s credit card. Some card companies, including Discover and American Express, act as their own issuing banks.

5   The issuing bank approves or declines the request. Depending on the card type, the bank could also use the Address Verification Service (AVS) to determine whether the customer provided the correct billing address. For more information about AVS, refer to AVS service documents via the CyberSource Services Documentation at <http://www.cybersource.com/support_center/support_documentation/services_documentation/payment.php> or as described in this integration guide.

6   CyberSource runs its own tests, then tells you if the authorization succeeded.

7   Response is sent back to the client.

### Taxes

Online Customer adds Product(s) to Cart and proceeds to Checkout.

As soon as shipping information is entered and validated, taxes are updated to reflect current tax rates based on six basic criteria:

1. Customer ship to address
2. Merchant ship from address
3. Merchant point of order origin (POO)
4. Merchant point of order acceptance (POA)
5. Product code
6. Merchant nexus

Product information is provided on an individual line item basis and all merchant/request IDs are captured for future reference. When the customer enters in shipping information, the Tax Service is called to calculate taxes.

### Address Verification Service (AVS)

AVS does not exist as a stand-alone callable service. Please refer to the Credit Card Authorization Service walkthrough for high level walkthrough.

### Delivery Address Verification Service (DAV)

DAV service may be run as a stand-alone callable service, as well as be performed as a part of other services. Please refer to Credit Card Authorization Service for more information regarding the DAV service, as an integral part of payment auth.

As a stand-alone service, the process is defined as:

* Customer enters shipping information
* Shipping information passes client-side validation (required elements filled in)
* Shipping information passes basic server-side validation (syntactically correct)
* Request is made to CyberSource DAV Service
* Response returns DAVReasonCode (100=Success)
* End node returns either: authorized, declined or error (authorized==success, declined==failure)
* Captured validation information is extracted from pipeline dictionary to present user with choices to correct problems, confirm “standardized” formatting or try again
* If service is successful, allow ShippingAddress save operation to continue

### Bill Me Later (BML)

A customer selects Bill Me Later during the checkout process at your site, similar to selecting the option to pay via Visa or MasterCard.

To request a Bill Me Later authorization, Demandware pipeline sends a request for a credit card authorization but instead of including a credit card number in the request, send the customer’s Bill Me Later account number set in the custom preferences.

To bill the customer, send a request for a credit card capture. No additional capture request fields are required for a Bill Me Later capture, unless you are processing multiple captures.

The Bill Me Later authorization service pipeline allows storefront application to request for credit authorization for the total order amount through the Bill Me Later.

The Demandware Cybersource–AuthorizeBML pipeline populates the authorization request with ship-to, bill-to, credit card data, and purchase total data from the basket and invokes the authorization web service call using CyberSource web service API.

Bill Me Later authorization sequence process:

* If it is the first time a customer has used Bill Me Later, they are presented with terms and conditions and asked for date of birth and last 4 digits of social security number.
* Once terms are agreed to, checkout continues as usual.
* A real-time credit decision is made and the consumer is notified within 3-5 seconds.
* On subsequent purchases, the customer simply chooses Bill Me Later, a real-time credit decision is made and the customer is notified within 3-5 seconds.
* In rare circumstances, additional verification steps are applied.
* After completion of the first purchase a welcome email is sent by Bill Me Later. This email contains a secure link for login to the self-service website, as well as a user ID for the customer.
* Your customer typically receives a bill (sent by Bill Me Later, Inc.) in their mailbox within fifteen days of the settlement posting to the customer’s account. The due date for this bill is 25 days from the date the bill is rendered.
* You submit the transaction for settlement upon shipment of goods or rendering the service—just as you do with a credit card purchase. Funding is received within the same timeframe as that of a purchase made with a credit card.

### Decision Manager

The CyberSoruce Decision Manager provides Merchant and ability to set business rules, provide case management, and Reporting.

The CyberSource Decision Manager Business rule engine allows Merchant to analyze the order data based on predefined or custom rules. The business rules can be set by orders, by category, or by SKU. The

The Demandware CyberSource Cartridge pipeline processes incoming Decision Manager Request and set contained orders to the new decision made through the decision manager.

Entry point into Demandware pipeline:

http://<sandboxname>/on/demandware.store/Sites-<store>-Site/default/Cybersource-NewDecision?content=<xml content>

Sample incoming reviewed order status update xml file

<?xml%20version=”1.0”%20encoding=”utf-8”?>

<!DOCTYPECaseManagementOrderStatus SYSTEM “https://ebctest.cybersource.com/ebctest/reports/dtd/cmorderstatus\_1\_1.dtd”>

<CaseManagementOrderStatus

MerchantID=”sample\_merchant”

Name=”Case Management Order Status”

Date=”2008-12-18 12:22:09 GMT”

Version=”1.1”

nxmlns=”http://reports.cybersource.com/reports/cmos/1.0”>

<Update MerchantReferenceNumber=”10679256010963322294714” RequestID=”1744185012770167904567”>

<OriginalDecision>REVIEW</OriginalDecision>

<NewDecision>ACCEPT</NewDecision>

<Reviewer>sample\_reviewer</Reviewer>

<ReviewerComments>sample\_comment</ReviewerComments>

<Queue>sample\_queue</Queue>

<Profile>test</Profile>

<FollowonResult>

<Status>Success</Status>

<Application>Credit%20Card%20Settlement</Application>

<RequestID>1744185012770167904567</RequestID>

<Decision>Accept</Decision>

<ReasonCode>100</ReasonCode>

<Rcode>1</Rcode>

<Rflag>SOK</Rflag>

<RMsg>Request%20was%20processed%20successfully.</RMsg>

</FollowonResult>

</Update>

</CaseManagementOrderStatus>

### Payment Tokenization

Tokenization is the replacement of sensitive data with a unique identifier that cannot be mathematically reversed. In your environment, tokens take the place of sensitive credit card data. Typically, the token will retain the last four digits of the card as a means of accurately matching the token to the payment card owner. The remaining numbers are generated using proprietary tokenization algorithms.

How It Works

* To make a purchase on your website, the customer will enter their payment card information into the designated payment fields on the order page. These payment fields will be hosted by CyberSource using [Hosted Payment Acceptance](http://www.cybersource.com/products_and_services/payment_security/hosted_payment_acceptance/). When the customer hits the ‘submit’ button, the data is immediately encrypted and transmitted directly to CyberSource for storing, processing, and token generation. The payment data never enters your environment.
* The encrypted primary account number (PAN) is decrypted when it enters CyberSource’s Level 1, PCI-compliant data vault, where it is securely stored. The payment data is then passed on to the processing channel (bank) and returned to CyberSource with an accepted or denied result.
* CyberSource returns the result to you but substitutes the PAN data with a uniquely generated token. You store the token in your database of record system (such as ERP system) for future transactions or chargeback resolution on that account. Customer service representatives can easily verify customers as the custom token will retain the last four digits of the original PAN.

### Payer Authentication

CyberSource Payer Authentication services enable you to add support to your web store for card authentication services, including Visa Verified by VisaSM, MasterCard® and Maestro® SecureCode™ (UK Domestic and international), American Express SafeKeySM, and JCB J/Secure™.

These card authentication services deter unauthorized card use and protect you from fraudulent chargeback activity referred to as *liability shift*.

How It Works

Payer Authentication provides the following services:

* + **Check Enrollment**: Determines whether the customer is enrolled in one of the card authentication programs.
  + **Validate Authentication**: Ensures that the authentication that you receive from the issuing bank is valid.



The Check Enrollment service determines whether the customer is enrolled in one of the

Card authentication services:

* **No**: If the card is not enrolled, you can process the authorization immediately.
* **Yes**: If the card is enrolled, the customer’s browser displays a window where the customer can enter the password associated with the card. This is how the customer authenticates their card with the issuing bank.
* If the password matches the password stored by the bank, you need to verify that the information is valid with the Validate Authentication service. If the identity of the sender is verified, you can process the payment with the Card Authorization service.
* If the password does not match the password stored by the bank, the customer may be fraudulent. You must refuse the card and can request another form of payment.

### Full Authorization Reversal

A full authorization reversal is a follow-on transaction that uses the request ID returned from a previous authorization. The request ID links the full authorization reversal to the authorization. CyberSource uses the request ID to look up the customer’s billing and account information from the original authorization, which means merchant is not required to include those fields in full authorization reversal request.

## Use Cases Scenarios

### Credit Card Authorization

The following table outlines the possible Demandware actions based on the response of the CyberSource gateway. Each client may choose to handle the response code differently. As of release 2.10, all errors logged as “fatal”, can activate an email alert to recipients identified in business manager.

|  |  |  |  |
| --- | --- | --- | --- |
| **Response** | **DW Storefront Action** | **Cyber-**  **Source Code** | **CyberSource suggested response** |
|  |  |  |  |
| Successful transaction. | Continue Checkout | 100 |  |
|  |  |  |  |
| **Validation Errors** | | | |
| Request is missing one or more fields | Should not occur as validation should catch this Show user “denied” error message Log fatal error (email alert) | 101 | See the reply fields missingField\_0...N for which fields are missing. Resend the request with the complete information. |
|  |  |  |  |
|  |  |  |  |
| One or more fields in the request contains invalid data. | Should not occur as validation should catch this Show user “denied” error message Log fatal error (email alert) | 102 | See the reply fields invalidField\_0...N for which fields are invalid. Resend the request with the correct information. |
|  |  |  |  |
| **System Errors** | | | |
| General system failure. | Show user “Unable to process – Call Cust Service” error message Log fatal error (email alert) | 150 | Wait a few minutes and resend the request. |
|  |  |  |  |
|  |  |  |  |
| The request was received but there was a server time-out. | Show user “Unable to process – Call Cust Service” error message Log fatal error (email alert) | 151 | Wait a few minutes and resend the request. |
|  |  |  |  |
|  |  |  |  |
| The request was received but there was a service time-out. | Show user “Unable to process – Call Cust Service” error message Log fatal error (email alert) | 152 | Wait a few minutes and resend the request. |
|  |  |  |  |
| The request just wait and then timeout, ends up as exception on the Demandware script | This could be one of the unique scenario where CyberSource waits for the Merchant’s bank to authorize the order and exceeds timeout sets at the Demandware. This ends up into SOAP exception. Client code can handle this scenario differently. | Script sets Rason Code to 999 | Handle at client’s end depending on business rules associated with this scenario. |
|  |  |  |  |
| **Authorization denied errors** | | | |
| Declined the request because the card has expired. | Show user “Auth denied” error message | 202 | Request a different card or another form of payment. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| The account number is invalid. | Show user “Auth denied” error message | 231 | Request a different card or other form of payment. |
|  |  |  |  |
| **Gateway Account problem** | | | |
| There is a problem with your merchant configuration. | Show user “Unable to process – Call Cust Service” error message Log fatal error (email alert) | 234 | Do not resend the request. Contact Customer Support to correct the configuration problem. |
|  |  |  |  |
| **Fraud Management** | | | |
| The fraud score exceeds your threshold. | Show user “Unable to process – Call Cust Service” error message Log fatal error (email alert) | 400 |  |
|  |  |  |  |
| The order is marked for review by Decision Manager. | Proceed with checkout Leave DW order “unconfirmed” | 480 |  |
|  |  |  |  |
| The order is rejected by Decision Manager. | Show user “Unable to process – Call Cust Service” error message Log fatal error (email alert) | 481 |  |

### Taxes

List of use cases and appropriate action taken listed below:

|  |  |
| --- | --- |
| **Use case scenarios** | **Result** |
| If shipping information is specified, then a request is made to the Tax Service | If successful, the contents of the Basket are taxed and price totals are adjusted.  If failed, because of service outage or failed address verification then don’t update anything. Other services must handle AVS/DAV/Service outages before successful checkout and final sales tax calculation. Failure is logged for email notification. |
| Since cybersource charges per request to the tax service, the cartridge has been modified to reduce the number of tax requests. Subsequent tax requests in the current session are only made to cybersource if the line item’s products id, quantity or price has changed or if the basket merchandise price total (including order level and product level), adjusted shipping price totals or adjusted basket total price has changed. | If the basket state that would affect tax has changed then a tax call will be made to cybersource and the basket will be updated with the new tax prices.  If the basket state that would affect tax has not change, the request to cybersource is skipped. |

### Address Validation Service (AVS)

Note that AVS does not run as an independent process, but is instead an optional, integrated aspect of payment authorization. List of use cases and appropriate action taken listed below:

|  |  |
| --- | --- |
| **Use case scenarios** | **Result** |
| AVS Ignore Result set to true | AVS Information is captured, but does not affect authorization response. |
| AVS Ignore Result set to false | AVS information is captured and if result from AVS is error or declined, then propagates that result up to the calling service. |
| AVS Ignore Result is set to false & AVS Decline Flags is defined | Seed request with additional result codes which should also result in a declined response. |

### Delivery Address Verification Service (DAV)

List of use cases and appropriate action taken listed below:

|  |  |
| --- | --- |
| **Use case scenarios** | **Result** |
| DAV Enable is set to false | No DAV information will be requested. No correction/validation information will be collected from the response. |
| DAV Enable is set to true, DAV On Failure set to REJECT | DAV information will be requested from the calling service. DAV related corrections and validation information is captured, and a DAV-related failure will be propagated to the calling service. |
| DAV Enable is set to true, DAV On Failure set to APPROVE | DAV information will be requested from the calling service. DAV related corrections and validation information is captured, but the result does not affect Authorization result. |

### BML

List of use cases and appropriate action taken listed below:

|  |  |
| --- | --- |
| **Use case scenarios** | **Result** |
| BML Authorization failed with response.decision = ERROR | Client application to display appropriate user friendly message to the end user. |
| BML Authorization failed with response.decision = ACCEPT | Pipeline sets the Authorization code to BMLPaymentInstrument.paymentTransaction.transactionID and ends with Authorized status  Order object is populated with cybersource transaction ID |
| BML Authorization failed with script error, or exception | Pipeline ends with error status, client code to display appropriate error message to the end user. |

### Decision Manager

Updates order status with the new decision set through the Decision Manager. The order status is updated in Demandware through the incoming xml. There are following possible options:

The new order status can be set to either accepted or rejected. TheCybersource-NewDecision pipeline retrieves the order for the incoming XML content, read order number from the XML, and updates corresponding storefront order with the status passed in the XML for the order.

List of use cases and appropriate action taken listed below:

|  |  |
| --- | --- |
| **Use case scenarios** | **Result** |
| Incoming order status is set to “ACCEPT” | Read order from the order table; Update the status and set that add it to the accepted orders collection.  The accepted order collection can then be used to log and alert.  Sets the HTTP response code “200”. |
| Incoming order status is set to “REJECT” | Read order from the order table; update the status and set that add it to the declined orders collection.  The declined order collection can then be used to log and alert.  Sets the HTTP response code “200”. |

### Payment Tokenization

Payment Tokenization service stores the customer and card related sensitive data for future reuse. Updates order object with the subscription id received from CyberSource.

List of use cases and appropriate action taken listed below:

|  |  |
| --- | --- |
| **Use case scenarios** | **Result** |
| Create subscription response is set to “ACCEPT” | Place the order and update the order object with subscription id. |
| Create subscription response is set to “REJECT” | Place the order but leave the subscription field empty. Make entry in log files to record the event. |

### Payer Authorization

List of use cases and appropriate action taken listed below:

|  |  |
| --- | --- |
| Use case scenarios | Result |
| Enrolment Check Error | Merchant proceeds to authorization (optional) |
| Cardholder Not Participating | Merchant proceeds to authorization |
| Unable To Verify Enrolment | Merchant proceeds to authorization (optional) |
| Successful Authentication | Merchant proceeds to authorization |
| Authentication Failure | Merchant asks for another form of payment |
| Attempted Authentication | Merchant proceeds to authorization |
| Authentication Unavailable | Merchant proceeds to authorization (optional) |
| Invalid Authentication Response | Merchant asks for another form of payment |
| PARes Signature Error | Merchant asks for another form of payment |
| Whitespace in PARes | Merchant proceeds to authorization |

### V.me by Visa

V.me functionality can be added at Cart page and Payment page on a Demandware merchant site.

* CyberSource V.me Transaction Flow



**Step 1 :** Initiate Payment service request and reply—retrieves the apikey field and the signature

(token) field to be included in the v:init and v:buy tags.

**Step 2 :** V.me buy button functionality.

**Step 3 :** Get Checkout Details service request and reply—this service is optional and retrieves the

customers shipping information to calculate shipping and tax costs.

**Step 4 :** Confirm Purchase service request and reply—confirms the purchase total for the order.

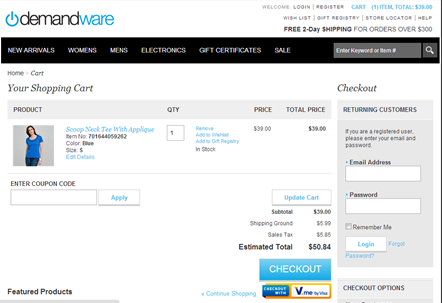
**Step 5 :** Authorization service request and reply—reserves the funds on the customer’s account.

**Step 6 :** Capture service request and reply—captures the funds that were reserved by an

authorization service request.

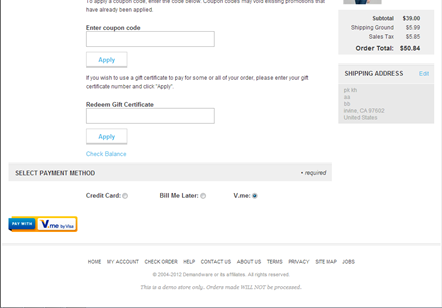
* Use Case 1 - Checkout using V.me on Cart page

Cart page after **“Checkout with V.me”** button is added on Demandware reference site SiteGenesis.

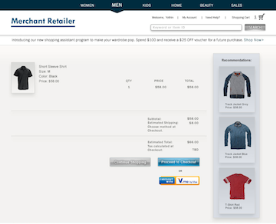
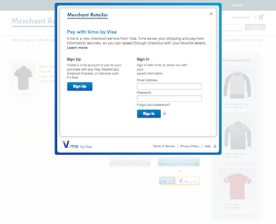
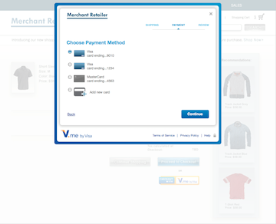
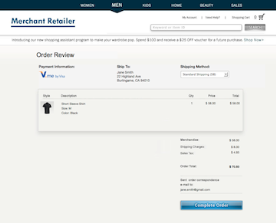


* Use Case 2 - Checkout using V.me on Payment page

Payment page after **“Pay with V.me”** button is visible on selecting radio button “V.me” on Demandware reference site SiteGenesis.

****

* User Flows on V.me payment widget.

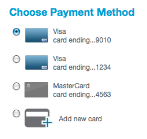


**Checkout with V.me Button (Merchant Site)**

**V.me Sign-up/Sign-in Lightbox**

**Payment Method Lightbox**

**Complete Order Page (Merchant Site)**



## Limitations, Constraints

Not currently implemented:

* Multiple shipments. Tax rates are only calculated for a single shipment per order. To implement tax service calculation for multiple shipments, a separate web service call must be made for each distinct “ship to” location.
* Custom User Interface components to correct address validation (DAV/AVS) errors and/or omissions or to confirm “standardized” address format corrections. All pertinent data is collected, but because each merchant will have customized specifications how to deal with such information (or use other 3rd party solutions to play the same role), no default user interface is provided.
* Custom user interface for view, update and delete subscription. All functionalities are created and working in stand-alone mode in **Cybersource\_Subscription.xml** pipeline. They have to customized and integrated as per the merchant specific needs.
* Custom user interface for Full Authorization Reversal. Full Authorization reversal is created and working in stand-alone mode in **Cybersource\_Services.xml** pipeline. It has to customized and integrated as per the merchant specific needs.

## Compatibility

This cartridge is supported under Demandware API release 12.6 and onward.

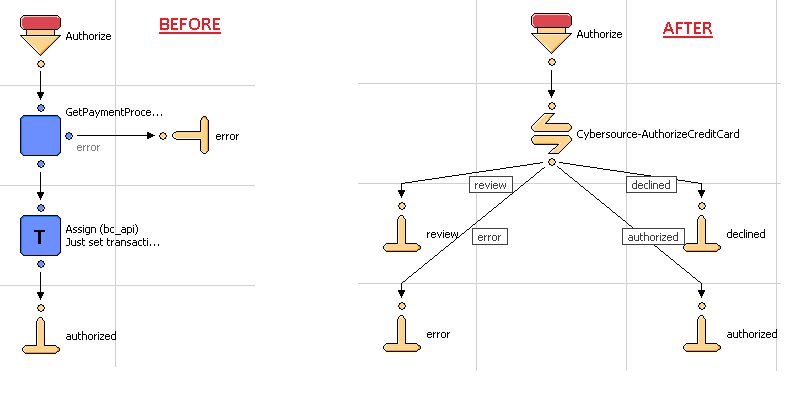
# Implementation Guide

## Custom Code

### Credit Card Auth

Update PPCybersourceCredit-Authorize pipeline to call Cybercource-AuthorizeCreditCard pipeline.

NOTE: Refer to the screen below for changes:



Alos, update HandlePayments pipeline to handle response code returned by CyberSource

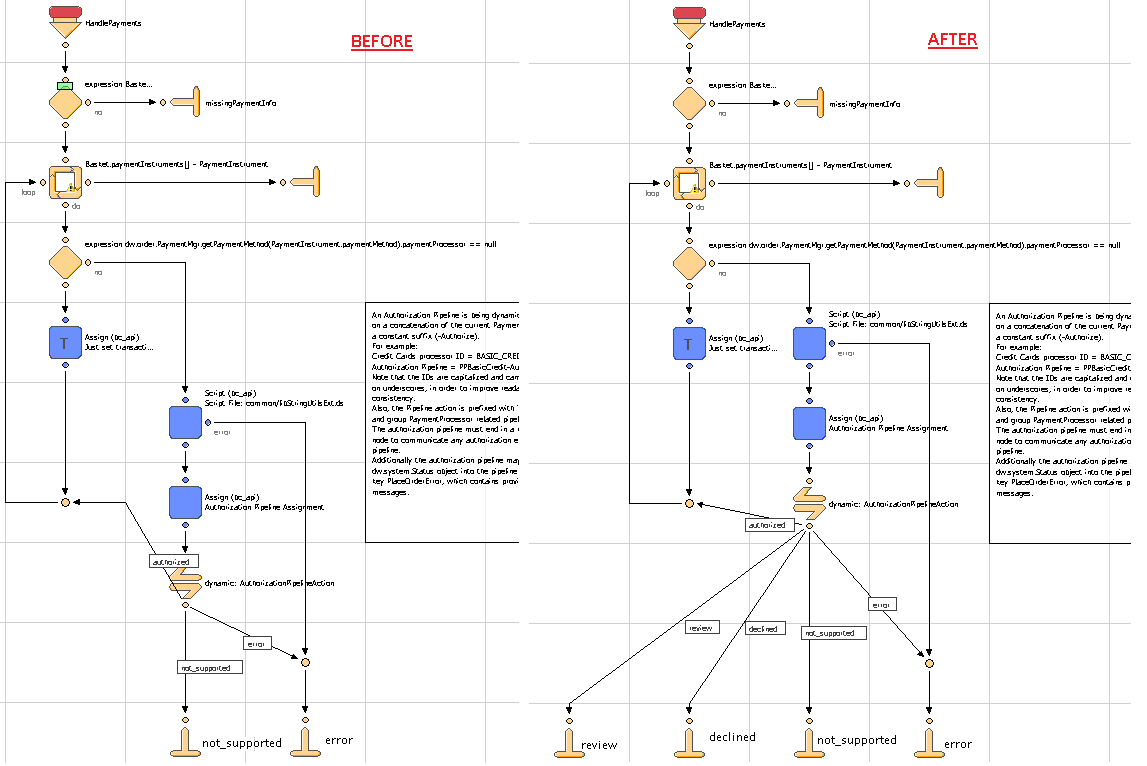
Authorized

Error

Declined and

Review

Note: Refer to screen below for changes:



### Bill Me Later

Update COPlaceOrder-HandlePayments pipeline to include Cybersource-AuthorizeBML.

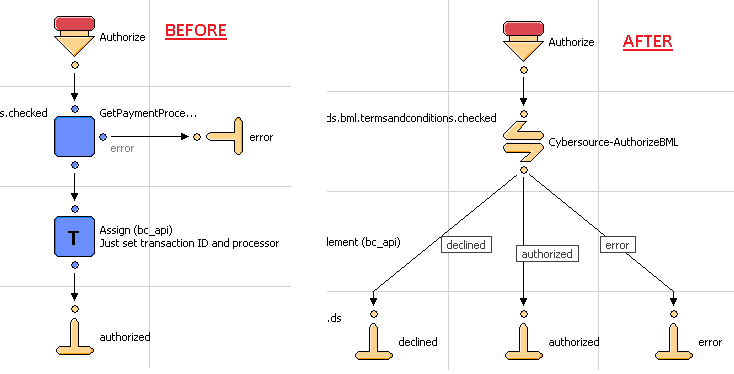
Add logic to handle following responses from the Cybersource-AuthorizeBML:

Authorized

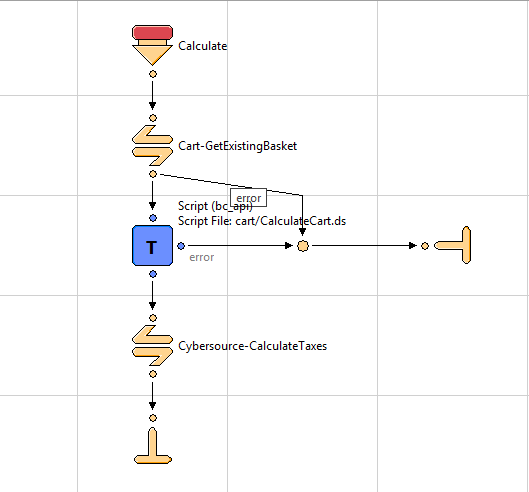
Error

Declined

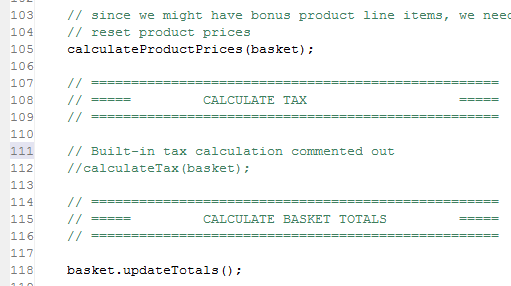
Note: Refer to screenshot below for changes:



### Tax Service

Update Cart-Calculate pipeline to run the Cybersource-CalculateTaxes call node after running the cart/calculateCart.ds script.  


Comment out the built in tax calculation function call in cart/calculateCart.ds



In order to avoid tax calculation call multiple times, set parameter SkipTaxCalculation to true in current session scope in the COShipping-UpdateShippingMethodList pipeline. Refer to the following screenshot:



  
Address Verification Service

Provide Site Preference values for 2 AVS-related business rules:

CsAvsIgnoreResult – Determines whether AVS failures will force an auth failure

CsAvsDeclineFlags –Determines how “correct” an address must be to produce a failure result

Augment UI interaction nodes to deal with AVS failure or correction confirmation dialogs, wherever Payment Authorization takes place, typically within the COPlaceOrder-Start and COSummary-Submit pipelines.

### Delivery Address Validation Service

Provide Site Preference values for 2 DAV-related business rules:

CsDavEnable – Determines whether DAV features are enabled for payment auth requests

CsDavOnAddressVerificationFailure –Determines whether a DAV failure will result in a payment auth failure

Augment UI interaction nodes to deal with AVS failure or correction confirmation dialogs, wherever Payment Authorization takes place, typically within the COPlaceOrder-Start and COSummary-Submit pipelines.

### Payment Tokenization Service

Provide Site Preference values for Payment Tokenization related business rules:

CyberSource – Enable Tokenization:– Determines whether payment tokenization service has to be enabled or not.

All functionalities related to CyberSource Payment Tokenization are created and working in stand-alone mode in Cybersource\_Subscription.xml pipeline. They have to customized and integrated as per the merchant specific needs

### Full Authorization Reversal

Full Authorization reversal is created and working in stand-alone mode in Cybersource\_Services.xml pipeline. It has to customized and integrated as per the merchant specific needs.

### Payer Authentication Service

Provide Site Preference values for 5 Payment Authorization related business rules:

CyberSource Merchant ID (PA):– Determines which CyberSource merchant id to be used for payer authentication. It can be same as default merchant id as well.

CyberSource Merchant Password (PA): –Password corresponding to the merchant account.

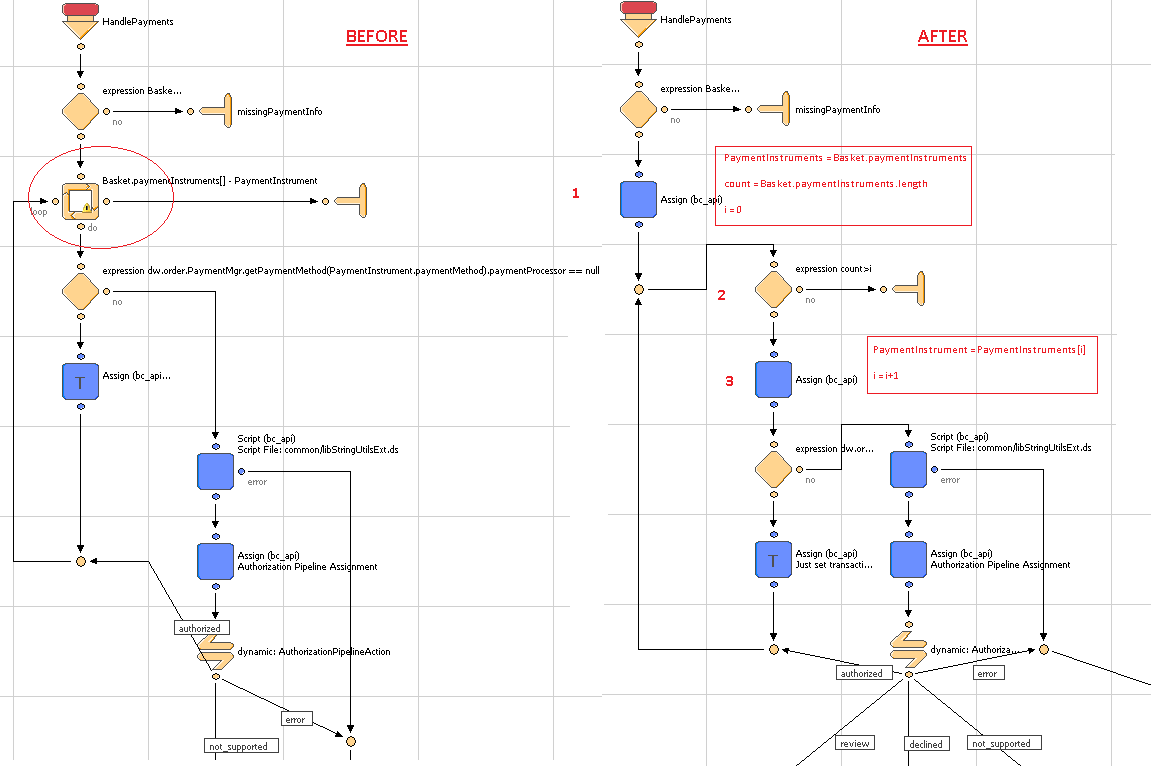
CyberSource Merchant Name (PA): –Merchant Name to be used for service.

CyberSource Save Proof.xml (PA): –Determines whether to save proof.xml (received from CyberSource response) as part of order object.

CyberSource Save ParesStatus (PA): –Determines whether to pass ParesStatus received as input parameter from Pa authorization request as input parameter to ccAuthorization request.

Update COPlaceOrder-HandlePayments pipeline to make custom loop to iterate through all the payment methods set in the basket.

NOTE: This is required as a result of a limitation of Loop Node due to which it does not work properly when an Interaction continue node is encountered in a loop.



In the diagram above, the loop node in the left side image is now replaced by a custom loop code, created using two Assign nodes and one Decision node. The configuration of Assign node is same as mentioned in the right side image.

### V.me by Visa

In order to integrate the V.me cartridge functionality merchant has to follow the instructions given in this section.

The cartridge code also includes sample files for reference. The changes made in sample files are denoted by appropriate **“BEGIN V.ME CHANGES” & “END V.ME CHANGES”** tags.

**Note:** It is suggested that merchant does all the changes defined in this section, to achieve the desired V.me functionality.

How it Works

In order to enable V.me buy button on website, following files have to be used from int\_visa\_VME cartridge.

* **visa\_buy.isml**

This file is used to make a reusable demandware module for V.me buy button. It has to be included in modules.isml, in order to be used as demandware reusable module. Once included, V.me buy button can be generated as follows:

***<isvisaBuy collect\_shipping="true" callback="vmeWidgetUIEventHandler"/>***

* + collect\_shipping parameter is used to map with V.me Collect Shipping variable.
    - True: Shipping details will be captured by V.me on payment widget
    - False: Shipping details will have to be provided by customer on merchant site.
  + Callback parameter is used to provide name of Javascript function used for receiving customer details from V.me widget.
* **visa\_init.isml**

This file is used to integrate v:init tag into Demandware template.

* **visa\_root.isml**

This file is used to create div required for V.me buy button. Also, it is used to include V.me proprietary javascript according to the environment (sandbox or production).

* **visa\_callback.isml**

This file is used to include javascript callback functions required by V.me buy button.

* **visa\_form.isml**

This file is used to include hidden html form Demandware template. Form is required to securely save V.me call-Id into Demandware order object.

Step1: Cartridge Import

Merchant needs to import both the cartridges **int\_visa\_VME & int\_cybersource** in Eclipse (with Demandware plugin) developer environment.

Step2: Enable V.me Payment Flow on Cart Page

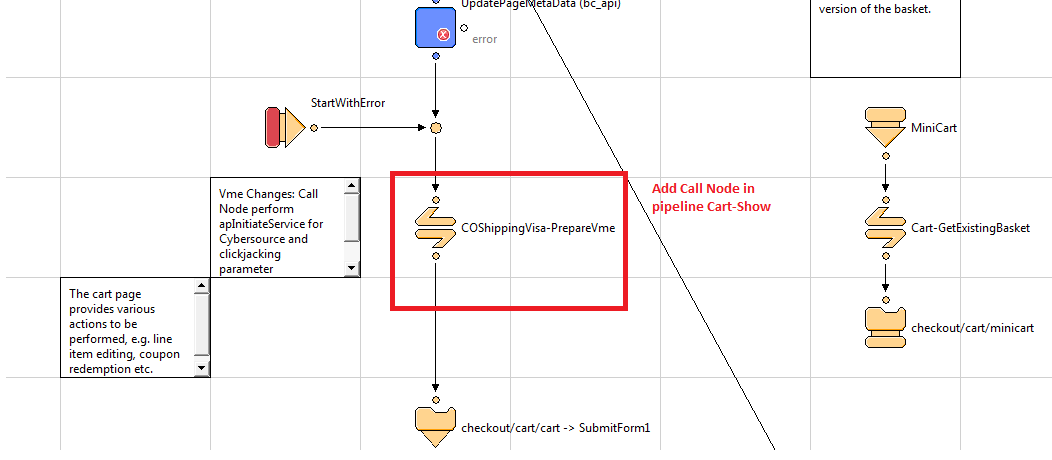
In order to enable V.me payment flow from Cart page, merchant is required to update following files:

* **Cart.xml**
  + **Sample File**

Cart\_sample.xml

* + **Changes Required**

1. Add Call Node (COShippingVisa-PrepareVme) in Cart-Show just before Interaction Continuation Node. Refer to the screenshot below:

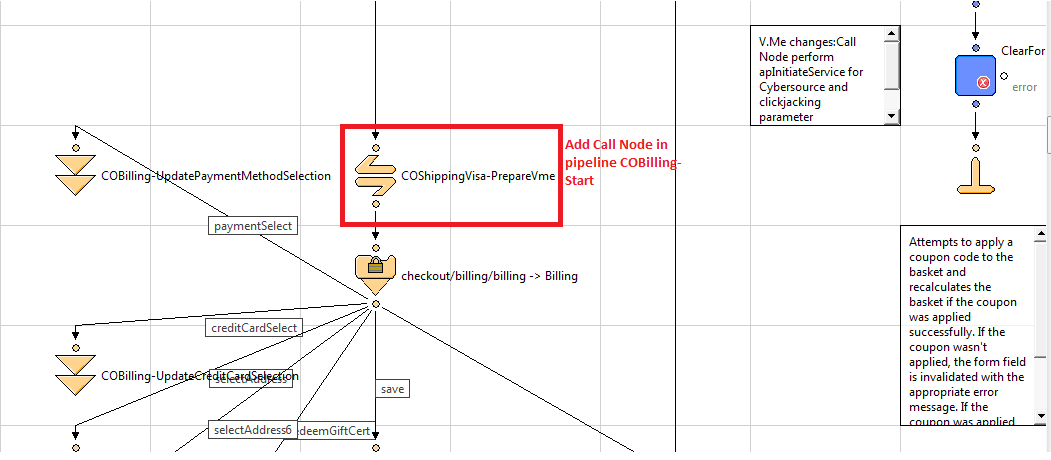
****

* **COBilling.xml**
  + **Sample File**

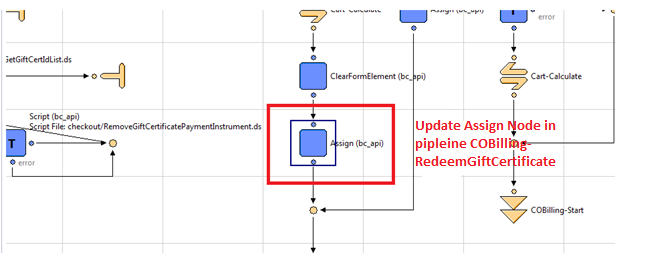
COBilling\_sample.xml

* + **Changes Required**

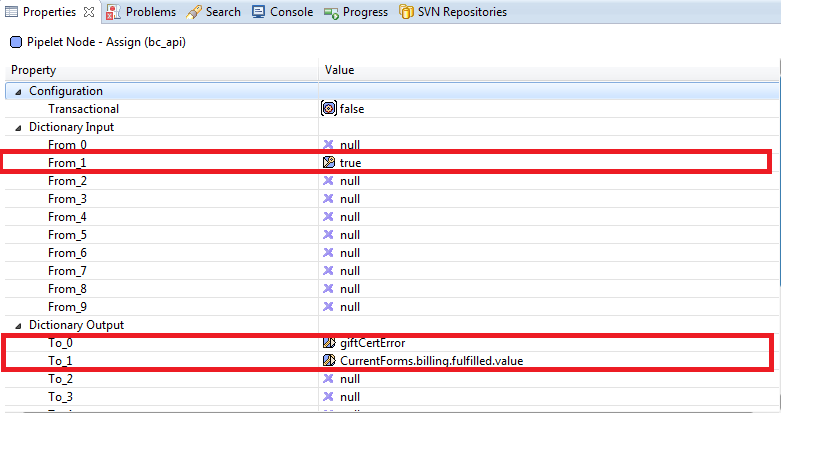
1. Add Call Node (COShippingVisa-PrepareVme) in COBilling-Start just before Interaction Continuation Node. Refer to the screenshot below:

****

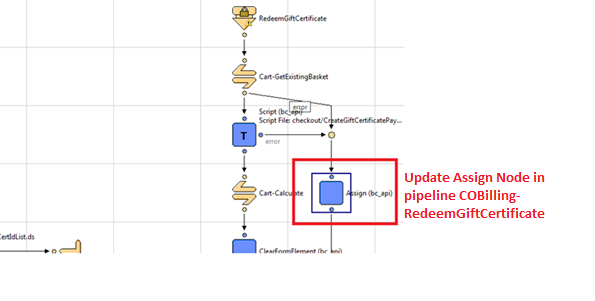
1. Update Assign Node in COBilling-RedeemGiftCertificate. Refer to the screenshot below:

****

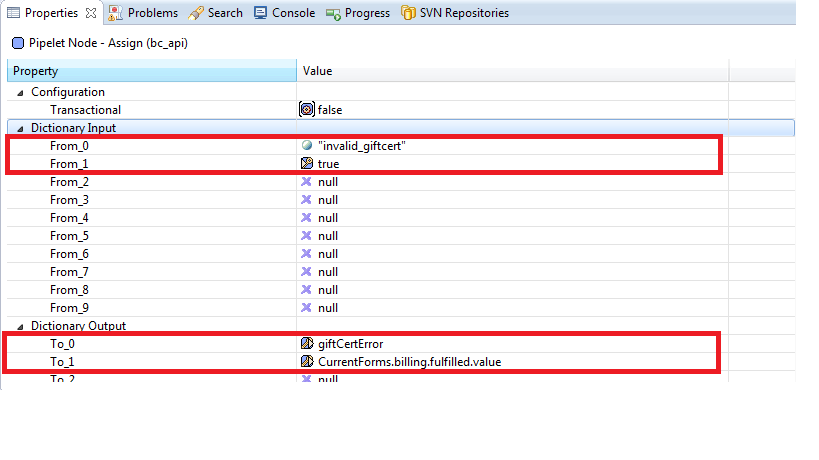
1. Set the properties for this Assign Node in the Properties View as shown below:

****

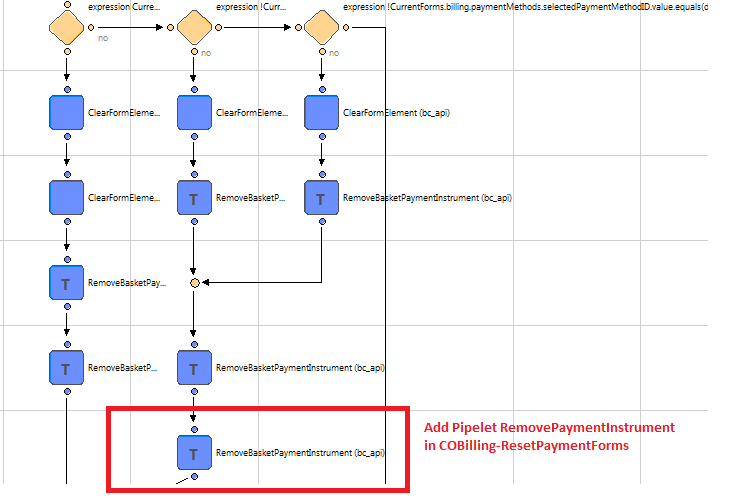
1. Update the second Assign Node in COBilling-RedeemGiftCertificate. Refer to the screenshot below

****

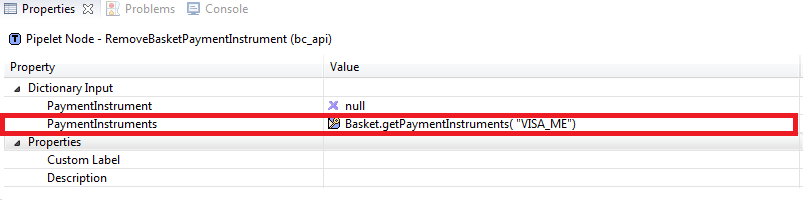
1. Set the properties for this Assign Node in Properties View as shown below

****

1. Add Pipelet (*RemoveBasketPaymentInstrument*) in COBilling-ResetPaymentForms

****

1. Set the properties for this Pipelet in Properties View.



* **modules.isml**
  + **Sample File**

modules\_sample.isml

* + **Changes Required**

1. Add following code block at the end of the file

**//BEGIN…………………………………………………………………**

*<iscomment>*

*Render the checkout with visa buy*

*</iscomment>*

*<isinclude template="visa\_modules.isml"/>*

**…………………………………………………………………….//END**

* **pt\_cart.isml**
  + **Sample File**

pt\_cart\_sample.isml

* + **Changes Required**

1. Add following code block just above the </head> tag

**//BEGIN…………………………………………………………………**

*<iscomment>Add template-specific header information here.</iscomment>*

*<iscomment>Style as part of clickjacking mechanism</iscomment>*

*<isif condition="${!empty(pdict.clickjackingflag)}">*

*<style> body { display : none;} </style>*

*</isif>*

*<isinclude template="visa\_callback.isml"/>*

**…………………………………………………………………….//END**

1. Add following code block immediately after <body> tag

**//BEGIN…………………………………………………………………**

*<isinclude template="visa\_init.isml" />*

**…………………………………………………………………….//END**

1. Add following code block immediately before </body> tag

**//BEGIN…………………………………………………………………**

*<isinclude template="visa\_root.isml"/>*

**…………………………………………………………………….//END**

* **cart.isml**
  + **Sample File**

cart\_sample.isml

* + **Changes Required**

1. Add following code block in the <div class = “cart-actions”>

**//BEGIN…………………………………………………………………**

*<div class="cart-action-continue-shopping" style="clear:both">*

*<isscript>*

*var enabled : Boolean = dw.order.PaymentMgr.getPaymentMethod("VISA\_VME").isActive();*

*</isscript>*

*<isif condition="${enabled}">*

*<isvisaBuy collect\_shipping="true" callback="vmeWidgetUIEventHandler"/>*

*</isif>*

*</div>*

**…………………………………………………………………….//END**

**Note:** It is suggested that merchant places the **isvisabuy** tag considering the DOM structure of their storefront cart page. Placement of **isvisabuy** tag will define the position of V.me buy button on the page.

1. Add following code block immediately before </isdecorate> tag

**//BEGIN…………………………………………………………………**

*<isinclude template="visa\_form.isml"/>*

**…………………………………………………………………….//END**

* **minibillinginfo.isml**
  + **Sample File**

minibillinginfo\_sample.isml

* + **Changes Required**

1. Add following code block after <isset name="paymentInstruments">

**//BEGIN…………………………………………………………………**

*<isset name="currentSite" value="${dw.system.Site.getCurrent()}" scope="page"/>*

*<isif condition = "${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value == 'VISA\_VME'}">*

*<div class = "mini-billing-address order-component-block">*

*<h3 class = "section-header">*

*<a href = "${URLUtils.https('COBilling-Start')}" class = "section-header-note"> ${Resource.msg('global.edit','locale',null)}*

*</a>*

*${Resource.msg('minibillinginfo.billingaddress','checkout',null)}*

*</h3>*

*<div class="details">*

*<img alt="Payment\_mark\_small" src="${currentSite.getCustomPreferenceValue('VmeAcceptLogo')}">*

*</div>*

*</div>*

*<iselse>*

**…………………………………………………………………….//END**

1. Add following code block at the end of condition <isif condition = "${!empty(billingAddress)}">

**//BEGIN…………………………………………………………………**

*</iselse>*

*</isif>*

**…………………………………………………………………….//END**

1. Replace the following code block

*<div class="mini-payment-instrument order-component-block <isif condition="${loopstate.first}"> first <iselseif condition="${loopstate.last}"> last</isif>">*

With the following code block

**//BEGIN…………………………………………………………………**

*<div class="mini-payment-instrument order-component-block">*

**…………………………………………………………………….//END**

1. Replace the following code block

*<isif condition = "${loopstate.first}"><span>${Resource.msg('minibillinginfo.paymentmethod','checkout',null)}</span></isif>*

With the following code block

**//BEGIN…………………………………………………………………**

*<span>${Resource.msg('minibillinginfo.paymentmethod','checkout',null)}</span>*

**…………………………………………………………………….//END**

1. Add following code block before <div class="details"> of <isif condition="${!empty(paymentInstruments)}">

**//BEGIN…………………………………………………………………**

*<isif condition="${paymentInstr.paymentMethod!='VISA\_VME'}">*

**…………………………………………………………………….//END**

1. Add following code block after <div class="details"> of <isif condition="${!empty(paymentInstruments)}">

**//BEGIN…………………………………………………………………**

*<iselse>*

*<div class="details">*

*<img alt="Payment\_mark\_small" src="${currentSite.getCustomPreferenceValue('VmeAcceptLogo')}">*

*</div>*

*</isif>*

**…………………………………………………………………….//END**

* **orderdetails.isml**
  + **Sample File**

orderdetails\_sample.isml

* + **Changes Required**

1. Replace the following code

*<isminiaddress p\_address="${Order.billingAddress}"/>*

With the following code block

**//BEGIN---------------------------------------------------------------------------------**

*<isloop items="${Order.getPaymentInstruments()}" var="paymentInstr" status="piloopstate">*

*<isifcondition="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).ID =='VISA\_VME'}">*

*<img alt="Payment\_mark\_small" src="${dw.system.Site.current.getCustomPreferenceValue('VmeAcceptLogo')}">*

*<iselse/>*

*<isminiaddress p\_address="${Order.billingAddress}"/>*

*</isif>*

*</isloop>*

**---------------------------------------------------------------------------------------//END**

1. Replace the following code

*<div class="payment-type"><isprint value="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).name}" /></div>*

*<isminicreditcard p\_card="${paymentInstr}" p\_show\_expiration="${false}"/>*

*<div class="payment-amount">*

*<span class="label">${Resource.msg('global.amount','locale',null)}:*

*</span>*

*<span class="value"><isprint value="${paymentInstr.paymentTransaction.amount}"/>*

*</span>*

*</div><!-- END: payment-amount -->*

With the following code block

**//BEGIN---------------------------------------------------------------------------------**

*<isif condition="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).ID =='VISA\_VME'}">*

*<div class="payment-type"><img alt="Payment\_mark\_small" src="${dw.system.Site.current.getCustomPreferenceValue('VmeAcceptLogo')}"></div>*

*<iselse/>*

*<div class="payment-type"><isprint value="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).name}" /></div>*

*<isminicreditcard p\_card="${paymentInstr}" p\_show\_expiration="${false}"/>*

*<div class="payment-amount">*

*<span class="label">${Resource.msg('global.amount','locale',null)}:*

*</span>*

*<span class="value"><isprint value="${paymentInstr.paymentTransaction.amount}"/></span>*

*</div><!-- END: payment-amount -->*

*</isif>*

**-------------------------------------------------------------------------------------//END**

* **orderdetailsemail.isml**
  + **Sample File**

orderdetailsemail\_sample.isml

* + **Changes Required**

1. Replace the following code

*<isminiaddress p\_address="${Order.billingAddress}"/>*

With the following code block

**//BEGIN---------------------------------------------------------------------------------**

*<isloop items="${Order.getPaymentInstruments()}" var="paymentInstr" status="piloopstate">*

*<isifcondition="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).ID =='VISA\_VME'}">*

*<img alt="Payment\_mark\_small" src="${dw.system.Site.current.getCustomPreferenceValue('VmeAcceptLogo')}">*

*<iselse/>*

*<isminiaddress p\_address="${Order.billingAddress}"/>*

*</isif>*

*</isloop>*

**--------------------------------------------------------------------------------------------//END**

Step3: Enable V.me Payment Flow on Payment Page

In order to enable V.me payment flow from Payment page, merchant is required to update following files:

* **pt\_ checkout.isml**
  + **Sample File**

pt\_checkout\_sample.isml

* + **Changes Required**

1. Add following code block just above the </head> tag

**//BEGIN…………………………………………………………………**

*<iscomment>Add template-specific header information here.</iscomment>*

*<iscomment>Style as part of clickjacking mechanism</iscomment>*

*<isif condition="${!empty(pdict.clickjackingflag)}">*

*<style> body { display : none;} </style>*

*</isif>*

*<isinclude template="visa\_callback.isml"/>*

**…………………………………………………………………….//END**

1. Add following code block immediately after <body> tag

**//BEGIN…………………………………………………………………**

*<isinclude template="visa\_init.isml" />*

**…………………………………………………………………….//END**

1. Add following code block immediately before </body> tag

**//BEGIN…………………………………………………………………**

*<isinclude template="visa\_root.isml"/>*

**…………………………………………………………………….//END**

* **billing.isml**
  + **Sample File**

billing\_sample.isml

* + **Changes Required**

1. Replace the following code

*<fieldset>*

With the following code block

**//BEGIN---------------------------------------------------------------------------------**

*<iscomment>block of code to display V.me logo with description</iscomment>*

*<isif condition="${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value==*

*'VISA\_VME'}">*

*<div id="vme-legend">*

*<legend>*

*Billing Address Provided By V.me*

*</legend>*

*<div>*

*<div class="details">*

*<img alt="Payment\_mark\_small" src="${dw.system.Site.current.getCustomPreferenceValue('VmeAcceptLogo')}">*

*</div>*

*<div></div>*

*</div>*

*</div>*

*</isif>*

*<fieldset id="billing-form">*

**----------------------------------------------------------------------------------//END**

1. Replace the following code

*<div class="form-row form-row-button">*

With the following code block

**//BEGIN------------------------------------------------------------------------------**

*<div class="form-row form-row-button" id="continue-button">*

**-----------------------------------------------------------------------------------//END**

1. Add following code before <isscript> tag

**//BEGIN------------------------------------------------------------------------------**

*<isinclude template="visa\_form.isml"/>*

**-----------------------------------------------------------------------------------//END**

* **paymentmethods.isml**
  + **Sample File**

paymentmethods\_sample.isml

* + **Changes Required**

1. Add the following code after <div id="PaymentMethod\_BML"> ends

**//BEGIN---------------------------------------------------------------------------------**

*<div id="PaymentMethod\_VISA\_VME" class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) &&*

*pdict.selectedPaymentID=='VISA\_VME'}">payment-method-expanded</isif>">*

*<isvisaBuy collect\_shipping="false" callback="vmeWidgetUIEventHandler" />*

*</div>*

**----------------------------------------------------------------------------------//END**

* **app.js**
  + **Sample File**

app \_sample.js

* + **Changes Required**

1. Add the following code in function changePaymentMethod(paymentMethodID) after line pmc.addClass("payment-method-expanded")

**//BEGIN---------------------------------------------------------------------------------**

*$cache.checkoutForm.find("#billing-form").show();*

*$cache.checkoutForm.find("#continue-button").show();*

*$cache.checkoutForm.find("#vme-continue-button").hide();*

*$cache.checkoutForm.find("#vme-legend").hide();*

**----------------------------------------------------------------------------------//END**

1. Add the following code in function changePaymentMethod(paymentMethodID) after condition if (paymentMethodID=="BML")

**//BEGIN---------------------------------------------------------------------------------**

*if(paymentMethodID==="VISA\_VME")*

*{*

*$cache.checkoutForm.find("#billing-form").hide();*

*$cache.checkoutForm.find("#continue-button").hide();*

*$cache.checkoutForm.find("#vme-continue-button").show();*

*$cache.checkoutForm.find("#vme-legend").show();*

*}*

**----------------------------------------------------------------------------------//END**

### Clickjacking Prevention

What is Clickjacking

Clickjacking (also known as user-interface redressing and IFRAME overlay) is an exploit in which malicious coding is hidden beneath apparently legitimate buttons or other clickable content on a website. It can be used for malicious actions, such as stealing confidential information, that are perpetrated against a user who is browsing a Web site. The user is “hijacked” by clicking a link on a contaminated Web page that executes the malware. The buttons may appear legitimate, but users are actually clicking buttons on a transparent layer they cannot see. The buttons can cause anything to happen, including making a purchase.

Prevention

Merchants must implement 2 mechanisms to ensure full protection from Clickjacking:

1. **Client side stylesheet/Javascript**

*<html>*

*<head>*

*...*

*<style id=”antiClickjack”>body{display:none;}</style>*

*<script type=”text/javascript”>*

*if (self === top) {*

*var antiClickjack = document.getElementById(“antiClickjack”);*

*antiClickjack.parentNode.removeChild(antiClickjack);*

*} else {*

*top.location = self.location;*

*}*

*</script>*

*...*

*</head>*

*<body>*

*...*

*</body>*

*</html>*

**Note:** The cartridge code already constitutes Client side code for Clickjacking prevention. Merchant does not need to perform any extra tasks in order to enable Client side Clickjacking prevention.

1. **Server Side**

Merchant must implement X-FRAME-OPTIONS with either “X-FRAME-OPTIONS DENY” OR “X-FRAME-OPTIONS SAMEORIGIN” on all pages wheretheV.me Payment Widget is hosted.

**Note:** Due to Demandware platform restriction, Server side code for Clickjacking prevention could not be included in the cartridge. Merchants are advised to raise a support ticket with Demandware Product Support team in order to get Server Side code enabled on Production environment.

Testing Clickjacking Prevention Code

Follow these steps to test your Clickjacking prevention code implementation:

1. Install a host http server and write a simple html page to iFrame the V.me customer.
   1. For host HTTP server, you can download and install Apache Tomcat (<http://tomcat.apache.org/index.html>)
   2. Create an HTML page, such as the following:

*<html>*

*<body>*

*<iframe src="https://www.yoursite.com/" width=100% height=100%>*

*<p>Your browser does not support iframes.</p>*

*</iframe>*

*</body>*

*</html>*

1. The iFrame src in the test HTML page should point to your customer’s page which hosts the V.me lightbox.
2. If the both options (frame busting script and X-Frame-options) are implemented correctly in your customer’s V.me page, you should observe the following:
   1. IE8+

You see “this content cannot be displayed in a frame” error message:



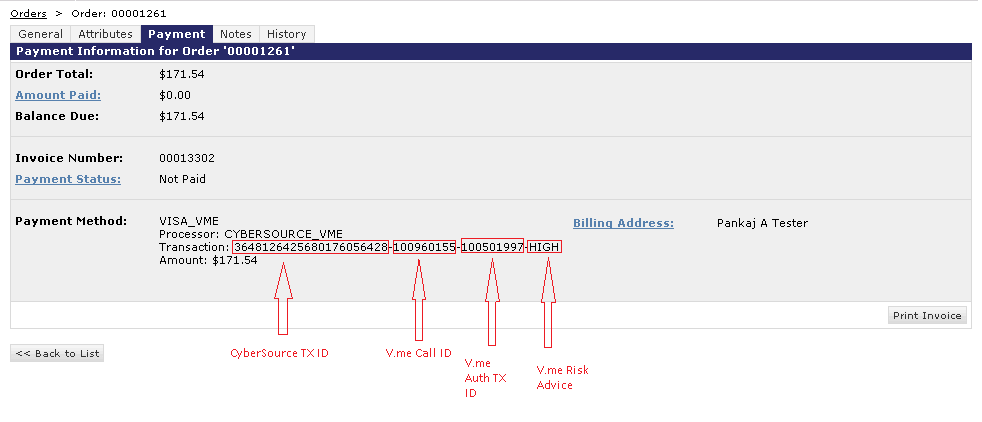
1. On other X-Frame-options supported browsers, you should see a blank page.
2. For all versions prior to the X-Frame-options supported versions, including IE7, your frame busting script should kick in to bust your customer site out of your hosted HTML page.

**Note:** You cannot put your site in an iFrame.

### Order Export XML – New Fields

After successful integration of the cartridge, Merchants can use the V.me button on Cart page and Payment page to process the Checkout flow.

On successful Order Checkout, using V.me as payment instrument, merchants can verify the V.me call Id and V.me Authorization ID by referring Order object created in Demandware Business Manager. Refer to the screen below for Order Object impact:



Note two important changes in Payment Tab of the Order Object:

1. **Processor:** CYBSERCOURCE\_VME
2. **Transaction:** This field will contain fields separated by delimiter ‘-‘. Format for this field is:

{Cybersource Tx ID} - {Call Id} - {Authorization Id}-{Risk Advice}

1. **Call Id -**Call Id generated by V.me payment Widget.
2. **Authorization Id** -Transaction ID returned by Visa server for Payment Authorization request for specific order.
3. **Risk Advice -**Risk Advice returned by Payment Authorization request for specific order.

Refer to the sample Order XML for new fields in Order Export

<?xml version="1.0" encoding="UTF-8" ?>

<orders xmlns="http://www.demandware.com/xml/impex/order/2006-10-31">

<order order-no="00001251">

<order-date>2013-03-26T09:38:25.627Z</order-date>

<created-by>storefront</created-by>

<original-order-no>00001251</original-order-no>

<currency>USD</currency>

<customer-locale>default</customer-locale>

<invoice-no>00013101</invoice-no>

<customer>

<customer-no>00007501</customer-no>

<customer-name>Pankaj A Tester</customer-name>

<customer-email>pankajtester9@gmail.com</customer-email>

<billing-address>

<first-name>Pankaj A</first-name>

<last-name>Tester</last-name>

<phone>2039121212</phone>

</billing-address>

</customer>

<status>

<order-status>OPEN</order-status>

<shipping-status>NOT\_SHIPPED</shipping-status>

<confirmation-status>CONFIRMED</confirmation-status>

<payment-status>NOT\_PAID</payment-status>

</status>

<current-order-no>00001251</current-order-no>

<product-lineitems>

<shipping-lineitems>

<shipments>

<totals>

<payments>

<payment>

<custom-method>

<method-name>VISA\_VME</method-name>

</custom-method>

<amount>227.84</amount>

<processor-id>CYBERSOURCE\_VME</processor-id>

<transaction-id>  3648126425680176056428-100960155-100501997-HIGH</transaction-id>

</payment>

</payments>

<remoteHost>14.140.116.135</remoteHost>

</order>

</orders>

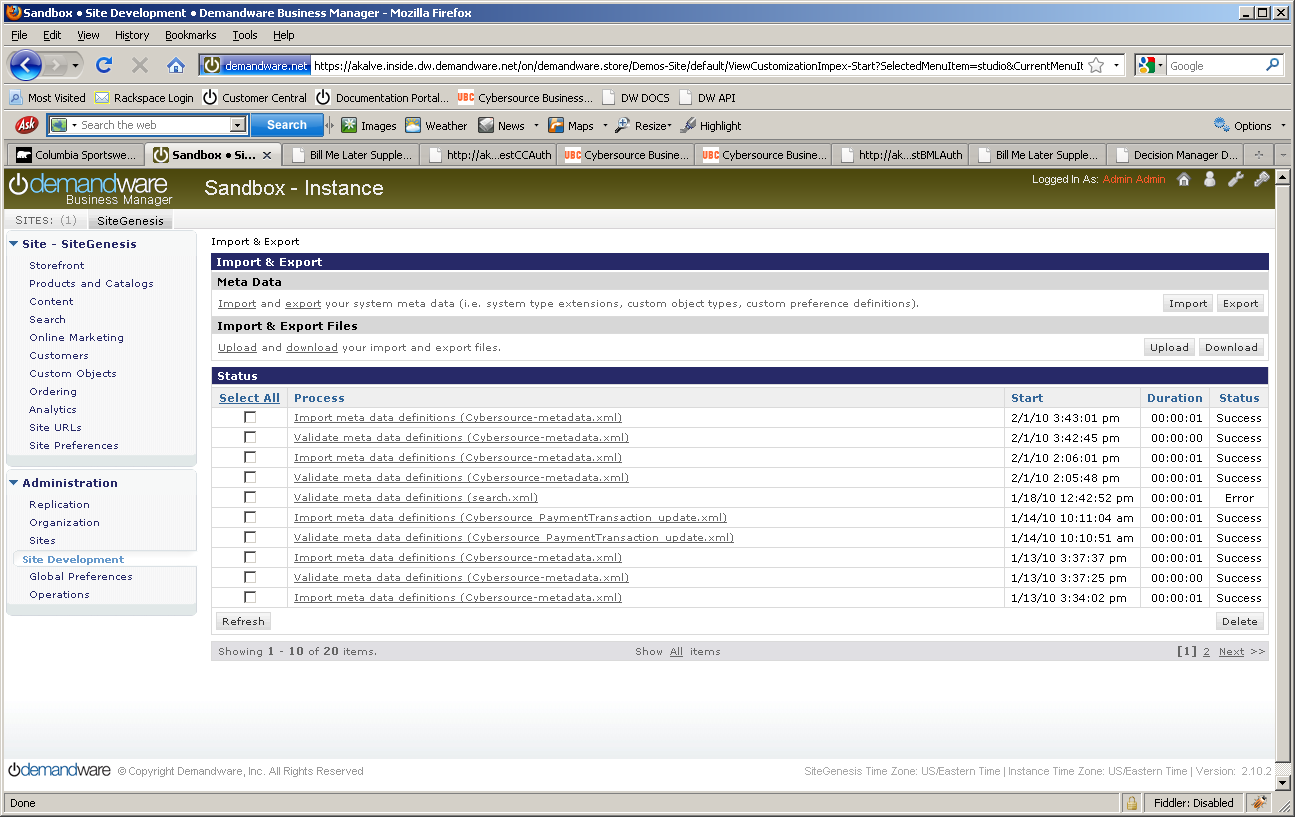
## Site Configuration

### Import Meta Data

Import following site configuration meta-data through Business Manager:

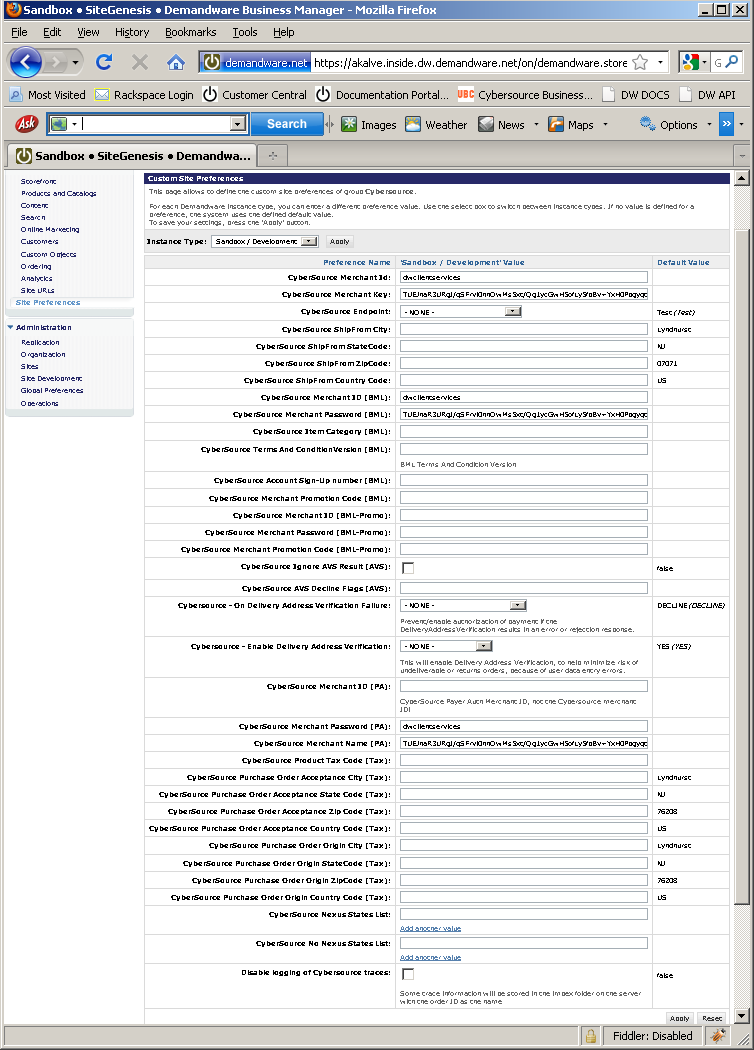
* /int\_cybersource/configuration/Cybersource-metadata.xml – sets all the site preferences
* /int\_cybersource/configuration/Cybersource\_PaymementTransaction\_updates.xml – add custom attributes to the Payment transaction object.
* /int\_cybersource/configuration/Vme\_Payment\_Method.xml – add new payment method as VISA\_VME
* /int\_cybersource/configuration/Vme\_Metadata.xml – set all site preferences for VISA\_VME

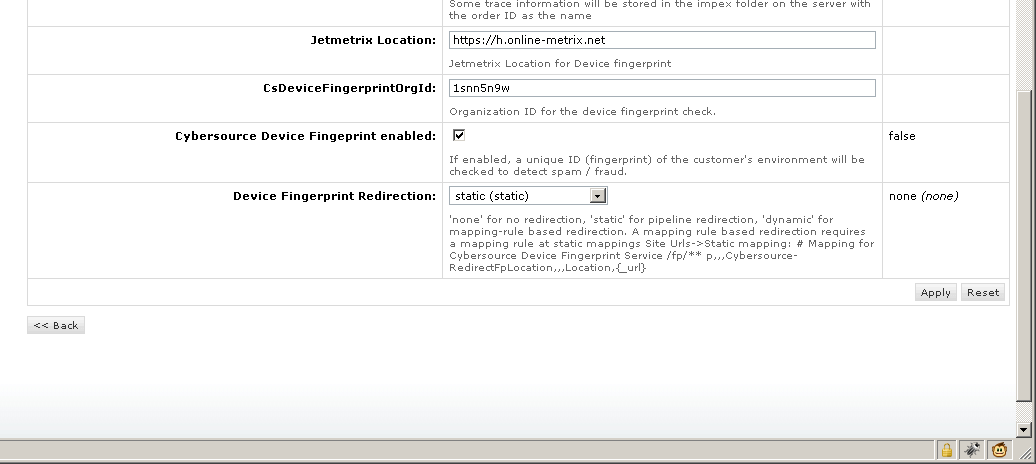
The following Business Manager Screenshot depicts the import / Export functionality:



### Configure Site Preferences

Update CyberSource site preference through Business Manager >StoreFront Site> Site Preferences.

The screen shot below depicts the site preferences configuration:….



Update credit card preference through Business Manager >StoreFront Site> Ordering> Payment Methods> Credit Card/Debit Cards > Enable Payer Authentication

The screen shot below depicts the site preferences configuration:



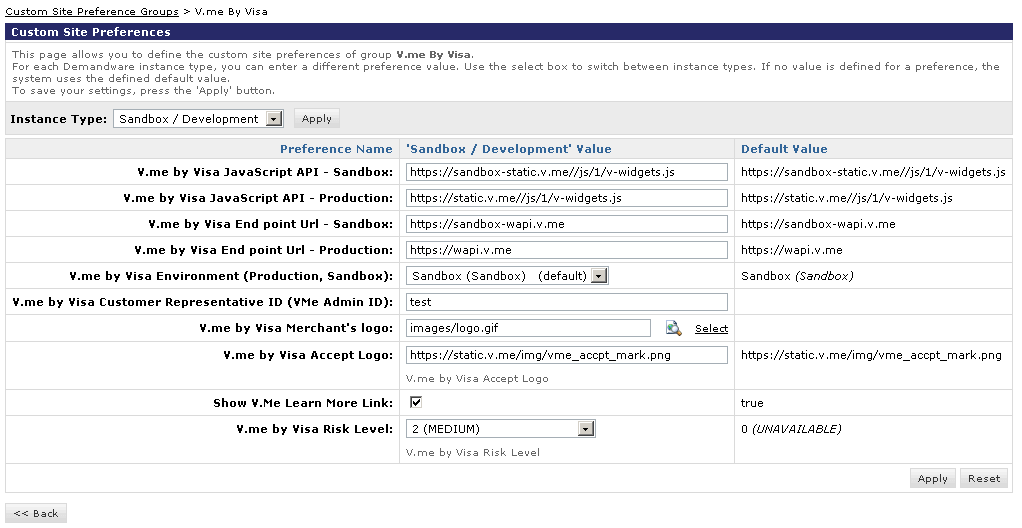
Update shipping method preference through Business Manager >StoreFront Site> Ordering> Shipping Methods > Name > CyberSource Shipping ID

The screen shot below depicts the site preferences configuration:



### Configure Site Preferences for V.me

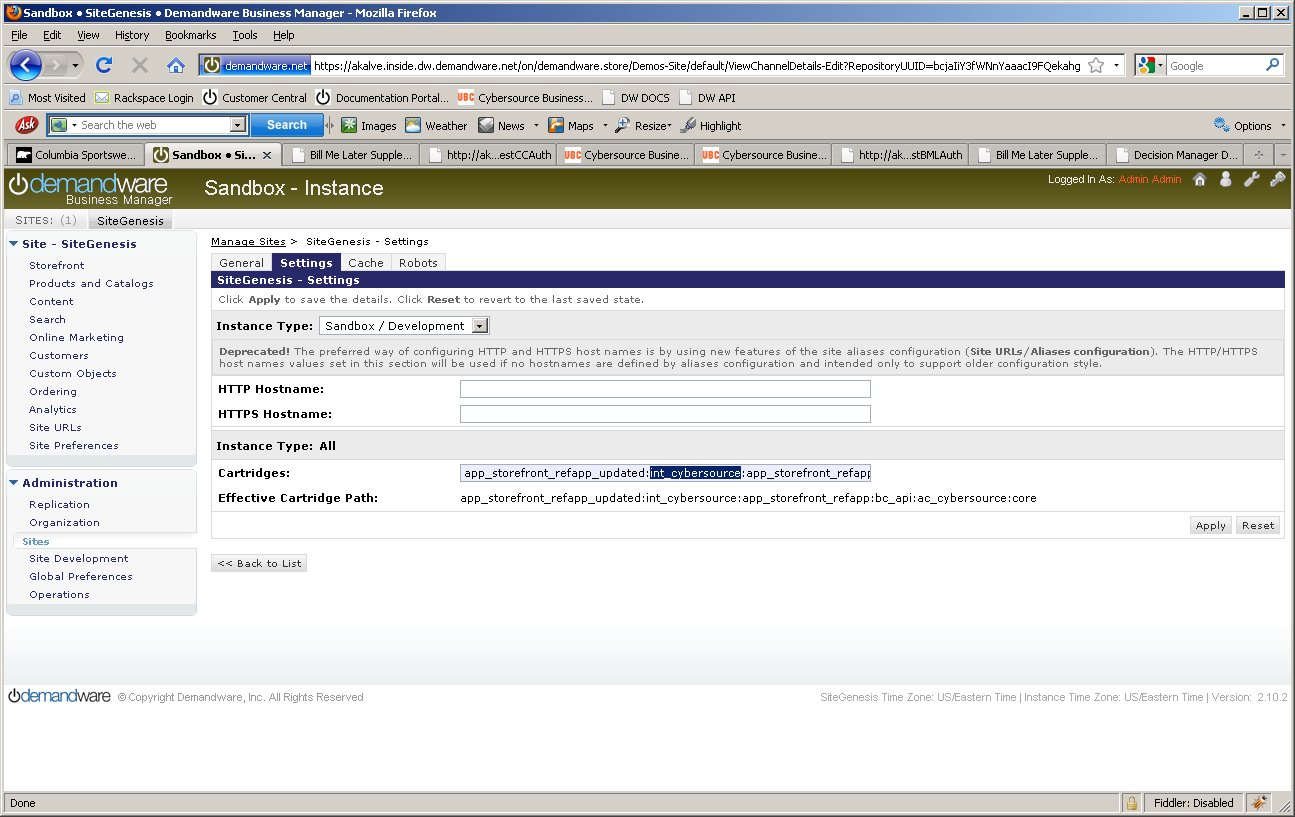
Verify VISA\_VME as newly added Custom Preferences Group



### Applying CyberSource Cartridge to the Site

Go to the “Administration” in the left hand list to expand the menu and select Sites > Manage Sites link. This will open a list of the active sites on the Demandware platform in your account. Click on the site for which you wish to add the CyberSource cartridge. This will open the General Settings page for that site.

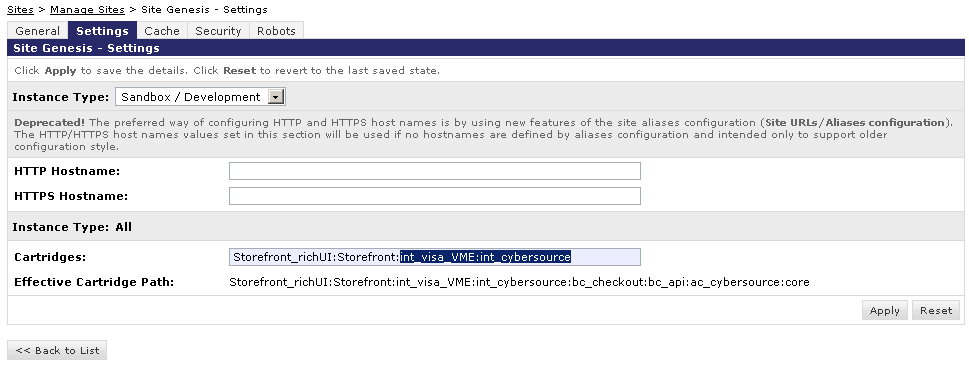
Add int\_cybersource cartridge to the cartridge path as depicted in the following screen:



### Applying V.me Cartridge to the Site

Go to the “Administration” in the left hand list to expand the menu and select Sites > Manage Sites link. This will open a list of the active sites on the Demandware platform in your account. Click on the site for which you wish to add the V.me cartridge. This will open the General Settings page for that site.

Add int\_visa\_VME cartridge to the cartridge path as depicted in the following screen:



## Testing

Use CybersourceUnitTest pipeline to test all the services as follows:

### Authorize Credit Card

Use and modify the CybersourceUnitTest-TestCCAuth pipeline and associated scripts and sub-pipelines. The end node of the unit test pipeline is a template which displays all relevant request/response information in an easy to digest manner. User can change static credit card and address data to observe various responses.

Note: Mark the start node as “PUBLIC” before executing the test case

### Tax Service

Use and modify the CybersourceUnitTest-TestTax pipeline and associated scripts and sub-pipelines. The script nodes for creating CreateCybersourceShipTo and CreateCybersourceBillTo objects have bindings to produce valid results, but otherwise can be manually modified to test against any domestic or international address.

The end node of the unit test pipeline for taxes is a template which displays all relevant request/response information in an easy to digest manner, to aid the debugging the various response codes and corrected address response.

Note: Mark the start node as “PUBLIC” before executing the test case

### Address Verification Service (AVS)

Use and modify the CybersourceUnitTest-TestCCAuth pipeline and associated scripts and sub-pipelines. By running simplified payment authorizations with different site preferences set, you can see how the AVS process works and how that result affects the overall payment authorization process.

Note: Mark the start node as “PUBLIC” before executing the test case

### Delivery Address Verification Service (DAV)

To test the stand-alone DAV service, use and/or modify the CybersourceUnitTest-TestDAVCheck pipeline and associated scripts and sub-pipelines. Like other test pipelines,test data can be customize to simulate various situations that need to be handled.

Note: Mark the start node as “PUBLIC” before executing the test case

The end node of the unit test pipeline for the stand-alone DAV Service is a template which displays all relevant request/response information in an easy to digest manner, to aid the debugging the various response codes and corrected address response.

Note: Mark the start node as “PUBLIC” before executing the test case

### Payment Tokenization

Use the Cybersource\_Subscription-Start pipeline to start Subscription creation test suite. By entering test data you can use the various Payment Tokenization related services like Create Subscription, View Subscription, Update Subscription, Delete Subscription, Use Subscription for One Time Payment.

Note: Mark the start node as “PUBLIC” before executing the test case

### Full Authorization reversal

Use the Cybersource\_Services-Start pipeline to start Authorization reversal test suite. By entering test data merchant can use the CyberSource Full Authorization Reversal service.

Note: Mark the start node as “PUBLIC” before executing the test case

### Device Fingerprint

Call the pipeline CybersourceUnitTest-TestFingerprint to test the device Fingerprint Service. A CreditCard Authorization is done and a device fingerprint will be additionally submitted.

Note: Mark the start node as “PUBLIC” before executing the test case

### Payer Authentication

Call the pipeline CybersourceUnitTest-TestPA to test the Payer Authentication Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Authorization

Call the pipeline CybersourceUnitTest-TestVmeAuth to test the V.me Authorization Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Get Checkout Details

Call the pipeline CybersourceUnitTest-TestVmeGetCheckoutDetails to test the V.me Get Checkout Details Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Get Transaction Details

Call the pipeline CybersourceUnitTest-TestVmeGetTransactionDetails to test the V.me Get Transaction Details Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Confirm Purchase

Call the pipeline CybersourceUnitTest-TestVmeConfirmPurchase to test the V.me Confirm Purchase Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Authorization Reversal

Call the pipeline CybersourceUnitTest-TestVmeAuthReversalRequest to test the V.me Authorization Reversal Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Capture

Call the pipeline CybersourceUnitTest-TestVmeCaptureRequest to test the V.me Capture Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Refund

Call the pipeline CybersourceUnitTest-TestVmeRefundRequest to test the V.me Refund Service.

Note: Mark the start node as “PUBLIC” before executing the test case

### V.me Initiate Request

Call the pipeline CybersourceUnitTest-TestVmeInitiateService to test the V.me Initiate Service.

Note: Mark the start node as “PUBLIC” before executing the test case

# Cartridges Structure and Reference

### Pipelines

##### Cybersource

This contains the nodes that make the actual call to CyberSource based on what is needed, cc auth, bml, tax or address verification.

##### CybersourceData

This contains the node that creates the objects needed for the CyberSource pipeline. Each node in the pipeline creates a needed object for each request. This is the pipeline that will be modified by customers, as each customer may need to pass different information based on their needs

##### CybersourceUnitTesting

Used for testing.

##### Cybersource\_Subscription

This contains the nodes that makes call to different Subscription related services. This is the pipeline that will be referred and modified by the merchants, as each merchant may need to pass different information based on their needs.

### Scripts

There is a JS object for each request data elements. List of DS objects used:

* Cybersource\_BillTo\_Object.ds
* Cybersource\_BML\_Object.ds
* Cybersource\_Card\_Object.ds
* Cybersource\_Item\_Object.ds
* Cybersource\_PurchaseTotals\_Object.ds
* Cybersource\_ShipFrom\_Object.ds
* Cybersource\_ShipTo\_Object.ds
* Cybersource\_TaxRequest\_Object.ds
* CreateCyberSourceBillToObject\_UserData.ds
* CreateCyberSourcePaymentCardObject\_UserData.ds
* CreateCyberSourcePurchaseTotalsObject\_UserData.ds

The following scripts are used in CyberSource pipeline to make the CyberSource web service calls.

* TaxationRequest.ds
* BMLAuthRequest.ds
* CCAuthRequest.ds
* DAVRequest.ds
* PayerAuthEnrollCheck.ds
* PayerAuthValidation.ds

The following scripts are used in Cybersource\_Subscription pipeline to make the Payment Tokenization web service calls.

* CreateSubscription.ds
* DeleteSubscription.ds
* OnDemandSubscription.ds
* UpdateSubscription.ds
* ViewSubscription.ds

The following scripts are used to create the objects needed. These are the scripts that customers would change to fit their needs

* CreateCybersourceBillToObject.ds
* CreateCybersourceBMLObject.ds
* CreateCybersourceBMLPaymentCardObject.ds
* CreateCybersourcePaymentCardObject.ds
* CreateCybersourcePurchaseTotalsObject.ds
* CreateCybersourceShipFromObject.ds
* CreateCybersourceShipToObject.ds
* CreateCybersourceTaxationItemsObject.ds
* CreateCybersourceTaxationPurchaseTotalsObject.ds
* CreateCyberSourceTaxRequestObject.ds

These are the pipelines that use the scripts from above and may have need storefront specific integration:

* CybersourceData:CreateBillTo
* CybersourceData:CreateShipTo
* CybersourceData:CreatePaymentCard
* CybersourceData:CreatePurchaseTotals
* CybersourceData: CreateShipFrom
* CybersourceData: CreateTaxItems
* CybersourceData: CreateTaxService

The following is a library script that is used by the Request scripts to build the XML request that is passed to Cybersource. This lib script contains integration independent code and it doesn’t need any storefront specific changes.

* libCybersource.ds

### Templates

* payerauthentication.isml
* payerauthenticationredirect.isml
* pt\_payerauthentication.isml
* Contains templates used by unit test pipeline

### Configuration Files

Contains two configuration file as follows:

* Cybersource\_PaymentTransaction\_update.xml

### V.me Cartridge Structure

Code for V.me functionality is structured in two different cartridges:

* **int\_visa\_VME**
  + This cartridge constitutes changes required to the Storefront level e.g. adding V.me Buy button, Pipelines to process V.me as payment instrument.

|  |  |
| --- | --- |
| **Component** | **Files** |
| **Pipelines** | *COShippingVisa*  *Hashing*  *VISA\_VME* |
| **Forms** | *vme.xml*  *vMeBilling.xml* |
| **Scripts** | *ValidateCartForCheckout.ds*  *CreatePaymentInstrument.ds*  *CreateShipmentShippingAddress.ds*  *libStringUtilsExt.ds*  *UpdateFormInformation.ds*  *Utils.ds*  *CheckoutDetailObject.ds*  *hash.ds*  *HashUtils.ds* |
| **Templates** | *hash.isml*  *shippingResponse.isml*  *visa\_buy.isml*  *visa\_callback.isml*  *visa\_form.isml*  *visa\_init.isml*  *visa\_modules.isml*  *visa\_root.isml* |
| **Properties File** | *int\_visa\_VME.properties* |
| **Sample Files** | *Cart\_sample.xml*  *COBilling\_sample.xml*  *billing\_sample.isml*  *minibillinginfo\_sample.isml*  *paymentmethods\_sample.isml*  *orderdetails\_sample.isml*  *orderdetailsemail\_sample.isml*  *modules\_sample.isml*  *pt\_checkout\_sample.isml*  *cart\_sample.isml*  *pt\_cart\_sample.isml*  *app\_sample.js* |

# Typical Project Plan

## Roles, Responsibilities

Typically most of the integration works is done by the backend developer. We expect that the person doing this integration is familiar with the web service, xml processing and has hands on experience with the Demandware platform.

## Typical Efforts and Timelines

The level of effort is mostly detected by the services merchant may choose from the CyberSource cartridge.

|  |  |  |
| --- | --- | --- |
| **CyberSource Service** | **Level of Effort (LOE)** | **Dependencies** |
| Initial Cartridge Setup | **0.5** – Person Day  List of tasks involved:   * Add CyberSource Cartridge to the project * Import Cybersource-metadata.xml * Import Cybersource\_PaymentTransaction\_update.xml | * Cartridge is available |
| Authorize Credit Card | **0.5** – Person Day  List of tasks involved:   * Integrate CyberSource-AuthorizeCreditCard pipeline with COPlaceOrder. | * Merchant ID and Key is established for the client. * Site Preferences for authorization configured with above ID and Key. |
| Device Fingerprint (as addition to Authorize Credit Card) | **0.5** Person Day | * Enable Device Fingerprint, set Organization ID * Add include at billing page. |
| Address Verification Service (AVS)\* | **0.5** – Person Day | * Initial Cartridge Setup |
| Delivery Address Verification (DAV)\* | **0.5** – Person Day | * Initial Cartridge Setup |
| Bill Me Later (BML) | **0.5** – Person Day | * Setup Account with Bill Me Later. |
| Decision Manager | **0.5** – Person Day | * Access to Decision Manager. * Business rules are defined. * Order status notification URL pointing to Cybersource-NewDecision pipeline is defined. |
| Payment Tokenization\* | **0.5** – Person Day  +  Depends on customization needs | * Initial Cartridge Setup |
| Payer Authentication | **1.5** – Person Day | * Initial Cartridge setup * Update CoPlaceOrder-HandlePayments pipeline * Handle error scenarios in merchant specific ways |
| V.me Initial Cartridge setup | **0.5** – Person Day | * Cartridge is available * Merchant account on CyberSource servers is setup for V.me integration |
| V.me integration on Cart Page | **1.0** – Person Day | * Initial Cartridge Setup |
| V.me integration on Payment Page | **1.0** – Person Day | * Initial Cartridge Setup |

\***Note that because customized user interface elements are completely dependent on merchant specification, the time required to interact with the customer to correct address information or confirm standardized address format corrections, is not included; only the time required to integrate with the web services is included, with minimal testing and simplified validation handling, i.e. Automatically make correction to a customer address, as per validation response.**

## Pre-Production Steps

In order to avoid misuse of unit testing pipelines on production instances it is advised to make following pipelines private before pushing code to production instances.

* CybersourceUnitTesting-TestBMLAuth
* CybersourceUnitTesting-TestCCAuth
* CybersourceUnitTesting-TestTax
* CybersourceUnitTesting-TestDAVCheck
* CybersourceUnitTesting-TestPA
* CybersourceUnitTesting-TestFingerprint
* Cybersource\_Subscription-Start
* Cybersource\_Subscription-CreateSubscription
* Cybersource\_Subscription-ViewSubscription
* Cybersource\_Subscription-UpdateSubscription
* Cybersource\_Subscription-DeleteSubscription
* Cybersource\_Subscription-OnDemandPayment
* Cybersource\_Services-Start
* Cybersource\_Services-Reversal
* CybersourceUnitTesting-TestVmeAuth
* CybersourceUnitTesting- TestVmeGetCheckoutDetails
* CybersourceUnitTesting- TestVmeGetTransactionDetails
* CybersourceUnitTesting- TestVmeConfirmPurchase
* CybersourceUnitTesting- StartVme
* CybersourceUnitTesting- TestVmeAuthReversalRequest
* CybersourceUnitTesting- TestVmeCaptureRequest
* CybersourceUnitTesting- TestVmeRefundRequest
* CybersourceUnitTesting- TestVmeInitiateService

# CyberSource Site Preferences

##### Site preference and description

|  |  |
| --- | --- |
| Site Preferences | Description |
| CyberSource Merchant Id(CsMerchantId) | CyberSource Merchant ID |
| CyberSource Merchant Key(CsSecurityKey) | CyberSource Security Key |
| CyberSource Endpoint(CsEndpoint) | CyberSource Web service End points:  Test <https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor>  Prod<https://ics2ws.ic3.com/commerce/1.x/transactionProcessor> |
| CyberSource ShipFrom City(CsShipFromCity) | Ship to data if fixed for the site |
| CyberSource ShipFrom StateCode(CsShipFromStateC)ode | Ship to data if fixed for the site |
| CyberSource ShipFrom ZipCode(CsShipFromZipCode) | Ship to data if fixed for the site |
| CyberSource ShipFrom Country Code(CsShipFromCountryCode) | Ship to data if fixed for the site |
| CyberSource Merchant ID(CsBmlMerchantId) | BML Merchant ID |
| CyberSource Merchant Password(CsBmlPassword | BML Merchant Key |
| CyberSource Item Category(CsBmlItemCategory) | BML Item Category |
| CyberSource Terms And ConditionVersionCsBmlTCVersion | BML Terms and Condition Version |
| CyberSource Account Sign-Up numberCsBmlNewAcctNo) | BML CyberSource account sign-up number |
| CyberSource Merchant Promotion Code(CsBmlPromoCode) | BML promo code |
| CyberSource Merchant ID(CsBmlPromoMerchantId) | BML promo merchant ID |
| CyberSource Merchant Password(CsBmlPromoMerchantPassword) | BML promo merchant key |
| CyberSource Merchant Promotion Code(CsBmlPromoPromoCode) | BML promo promo code |
| CyberSource Ignore AVS Result(CsAvsIgnoreResult) | AVS ignore results |
| CyberSource AVS Decline Flags(CsAvsDeclineFlags) |  |
| CyberSource – On Delivery Address Verification Failure(CsDavOnAddressVerificationFailure) |  |
| CyberSource – Enable Delivery Address Verification(CsDavEnable) | This will enable Delivery Address Verification, to help minimize risk of undeliverable or returns orders, because of user data entry errors. |
| CyberSource Merchant ID(CsPaMerchantId) | Payer Auth merchant ID |
| CyberSource Merchant Password(CsPaMerchantPassword) | Payer Auth Merchant Key |
| CyberSource Merchant Name(CsPaMerchantName) | Name |
| CyberSource Purchase Order Acceptance City(CsPoaCity) | CyberSource purchase order acceptance data – used by Tax |
| CyberSource Purchase Order Acceptance State Code(CsPoaStateCode) | CyberSource purchase order acceptance data – used by Tax |
| CyberSource Purchase Order Acceptance Zip Code(CsPoaZipCode) | CyberSource purchase order acceptance data – used by Tax |
| CyberSource Purchase Order Acceptance Country Code(CsPoaCountryCode) | CyberSource purchase order acceptance data – used by Tax |
| CyberSource Purchase Order Origin City(CsPooCity) | CyberSource purchase order origin data – used by Tax |
| CyberSource Purchase Order Origin StateCode(CsPooStateCode) | CyberSource purchase order origin data – used by Tax |
| CyberSource Purchase Order Origin ZipCode(CsPooZipCode) | CyberSource purchase order origin data – used by Tax |
| CyberSource Purchase Order Origin Country Code(CsPooCountryCode) | CyberSource purchase order origin data – used by Tax |
| CyberSource Nexus States List(CsNexus) | CyberSource nexus state list |
| CyberSource No Nexus States List(CsNoNexus) | CyberSource no nexus state list |
| Disable logging of CyberSource traces(CsDebugCybersource) | To enable/disable debugging |
| CyberSource Device Fingeprint enabled(CsDeviceFingerprintEnabled) | To enable / disable the device fingerprint for advanced fraud detection |
| Jetmetrix Location(CsJetmetrixLocation) | Location of device fingerprint service |
| CsDeviceFingerprintOrgId(CsDeviceFingerprintOrgId) | Id of DeviceFingerprintOrgId |
| Device Fingerprint Redirection(CsDeviceFingerprintRedirectionType) | None,static or dynamic for type of redirection. |
| CyberSource – Enable Tokenization(CsTokenizationEnable) | To enable/disable tokenization call in CC Authorization |
| CyberSource Save Proof.xml(CsPaEnableProofXML) | To enable/disable saving of proof.xml in order object |

# V.me Site Preferences

|  |  |  |
| --- | --- | --- |
| **Site Preference** | **Description** | **Default Value** |
| VmeSandboxJavaScriptAPI | Visa VMe Java Script API - Sandbox | https://sandbox-static.v.me//js/1/v-widgets.js |
| VmeJavaScriptAPI | Visa VMe Java Script API - Production | https://static.v.me//js/1/v-widgets.js |
| VmeEnvironment | Visa VMe Environment (Production, Sandbox) | Sandbox (Sandbox) |
| VmeCustomerRepresentativeID | Visa VMe Customer Representative ID (VMe Admin ID) | NA |
| VmeMerchantLogo | Visa VMe Merchant's logo | NA |
| VmeAcceptLogo | Visa VMe Accept Logo | https://static.v.me/img/vme\_accpt\_mark.png |
| VmeEnableLearnMoreLink | Show Visa VMe Learn More Link | true |

# Device Fingerprint

The device fingerprint enables CyberSource to detect fraud/spam more efficient.  
The device fingerprint can be used as an addition of the Credit Card Payment, it is not an independent service.

## How does it work?

During/before checkout three (invisible) ‘beacons’ at the checkout page (a javascript, an image and a flash object) would collect and transmit several client-specific parameters to CyberSource partner.

Those beacons contain the session Id.

With the Credit Card Payment, this session Id is transmitted again and CyberSource is able to combine the data for advanced fraud detection.

## Setup:

(Prerequisites: CyberSource cartridge is already installed).

1. Enable the device fingerprint at the Site Preferences of CyberSource and set the Organization ID (provided by CyberSource). The Merchant ID should be set already, anyway.
2. Include following snippet i.e. at the billing.isml page (Recommended: at bottom of page to have no visual impacts)

<isifcondition=*”${dw.system.Site.getCurrent().getCustomPreferenceValue(‘CsDeviceFingerprintEnabled’)}”*>

<isincludeurl=*”${URLUtils.url(‘Cybersource-IncludeDigitalFingerprint’)}”*/>

</isif>

Do a checkout with Credit Card payment. After this checkout, at the CyberSource Business Manager you will see (at the Transaction Manager):

*Device Fingerprint: submitted*

### Hints for the CsDeviceFingerprintRedirectionType:

To get improved deviceFingerprint results, CyberSource recommends redirecting the included code (loading a image, a flash and a javascript) pointing to the CsJetmetrixLocation, to a local domain.

There are three possible settings for this redirection: ‘none’, static’ and dynamic.

*None*no redirection, the beacons will be loaded direct from the CsJetmetrixLocation (i.e. https://h.online-metrix.net)

*Static* The beacons are included with a emandware pipeline call. The pipelinecall will redirect to the CsJetmetrixLocation.

Dynamic If set to dynamic, you have to specify a mapping rule at SiteUrls->Static Mappings.

All URLs matching the pattern will be redirected by the emandware Server.



Example for a matching mapping rule for the device fingerprint redirection

### Modified Scripts and pipelines for the device fingerprint

Scripts:

libCybersource .ds->addCCAuthRequestInfo modified.

CCAuthRequest.ds (updated WSDL reference)

TestCCAuth (Testcases plus added ‘response object’ dumping for developers)

Pipelines:

CybersourceUnitTesting (new test case)

# Known Issues

1. **In case of setting Ignore AVS Result custom preference to true, there can be a known issue as described below:**

**If the AVS response code received as N, the cartridge ignores the ccAuthReply reason code and processes the transaction under “review” status. This can lead to an ambiguous situation when the Credit Card was rejected, but due to the AVS code as “N”, the cartridge continued with order processing and successful order placement.**

# Supported Versions

This cartridge is compatible with following versions :

|  |  |
| --- | --- |
| **SiteGenesis** | Version 12.6 |
| **CyberSource API (WSDL/XML Schema)** | Version 1.80 |

# CyberSource document links

1. <http://www.cybersource.com/support_center/implementation/testing_info/simple_order_api/General_testing_info/soapi_general_test.html>
2. <http://www.cybersource.com/support_center/support_documentation/quick_references/view.php?page_id=422>
3. <http://apps.cybersource.com/library/documentation/dev_guides/CC_Svcs_SO_API/Credit_Cards_SO_API.pdf> - Page 163 - Appendix C.
4. <http://apps.cybersource.com/library/documentation/dev_guides/Getting_Started/Getting_Started_Advanced.pdf>
5. <http://www.cybersource.com/support_center/support_documentation/quick_references/>
6. <http://apps.cybersource.com/library/documentation/dev_guides/Payer_Authentication_IG/20090928_Payauth_IG.pdf>
7. <http://apps.cybersource.com/library/documentation/dev_guides/Payer_Authentication_IG/html/>
8. <http://apps.cybersource.com/library/documentation/dev_guides/CC_Svcs_IG_BML_Supplement/html/>
9. <http://apps.cybersource.com/library/documentation/dev_guides/Verification_Svcs_IG/20091012_Verification_IG.pdf>
10. <http://www.cybersource.com/support_center/support_documentation/services_documentation/tax.php>
11. <http://apps.cybersource.com/library/documentation/dev_guides/Tax_IG/Tax_Guide.pdf>
12. <http://apps.cybersource.com/library/documentation/dev_guides/Vme/Vme_SO_API.pdf>

# Release History

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 1.0.0.1 | 02/02/2010 | Initial release |
| 1.0.0.2 | 02/08/2010 | Device Fingerprint Feature added |
| 1.0.0.3 | 03/01/2012 | Updated Tax pipeline to remove unnecessary / redundant tax requests to reduce tax service charges. |
| 1.0.0.4 | 12/18/2012 | Updated Tax pipeline to remove redundant tax requests by using SkipTaxCalculation parameter |
| 1.1.0 | 01/16/2013 | Incorporated review comments from Demandware team |
| 1.1.0 | 02/06/2013 | Incorporated New changes as per new Site Genesis code |
| 1.2.0 | 03/28/2013 | V.me support changes added. |