int_virtualpiggy

Version 1.3.0

Table of Contents

1. Summary

- The component (**Oink** by Virtual Piggy) adds a possibility for merchant to enable COPPA compliant, Level 1 PCI certified checkout gateway designed for families and under aged minors. Oink integrates with the 3rd party payment gateways.

- The **Oink** component is free of charge. A contract is required between Virtual Piggy and a Merchant using the component. Virtual Piggy charges a percentage per successful transaction placed using the component. Please contact Virtual Piggy for the current rates.

- To use **Oink**, merchants must assign and install int_virtualpiggy cartridge, create custom system definitions, set **Oink** site preferences, create an **Oink** payment method for each site that will use cartridge, and provide callback URL's for **Oink**.

2.1 Functional Overview

The cartridge adds an opportunity to purchase goods using **Oink** (by Virtual Piggy©) payment provider. It contains:

- Authentication using **Oink** credentials
- Custom single-page checkout
- Credit Card authorization and capturing in both automatic and manual modes
- Confirmation email sending in case of order approval

2.2 Use Cases

The cartridge supports following use cases:

- Child checkout in automatic mode (no parent approval required)
- Child checkout in manual mode (parent approval required)
- Guest checkout (parent purchases for selected child)
- Automatic Capture (money get captured from CC automatically)
- Manual Capture (manual action for each approved order required)

2.3 Limitations, Constraints

- Single-shipment supported only
- One shipping method should be set for all **Oink** orders
- Gift Certificates are not supported
- Front-end integration was developed for SiteGenesis v12.5+. Solutions based on previous versions could require additional actions for integration.
- For Child checkout in manual mode (parent approval required) in the event that the parent rejects the order in the **Oink** interface the total amount of these orders will be added to the merchant's GMV total and result in charges with Demandware

2.4 Compatibility

Front-end is compatible with SiteGenesis 12.5. Back-end is compatible with API 2.10.6.

2.5 Privacy, Payment

Demandware has no access to the user of the cartridge credit card data (PCI Level 1 compliant) or a child's full name and address (COPPA compliant). Credit card data is stored within the Oink application.

For terms, please visit: http://www.oink.com/virtual-piggy-terms-service

For privacy policy, please visit: http://www.oink.com/virtual-piggy-privacy-policy

3. Implementation Guide

3.1 Setup

The int_virtualpiggy cartridge is required for the integration.

3.2 Configuration

The integration of **Oink** requires 5 Virtual Piggy configuration steps:

1. Create custom System Object definitions
2. Set Virtual Piggy Site Preferences for each site
3. Create a Virtual Piggy payment method for each site
4. Provide Callback URLs for Virtual Piggy
5. Assign the cartridge to a site

IMPORTANT: steps 1 and 3 could be performed at the same time and automatically using the "Site Import" feature. Detailed site import instructions are provided at the end of this section.

Import VP's System Objects

Select your **Site -> Site Development -> Import & Export**

Under **Import & Export Files** select **Upload** and upload **VPObjects.xml** (found in the **documentation** folder)

Return to **Site Development -> Import & Export**

Under **Meta Data,** select **Import.**

Select **VPObjects** from the file list and click **Next**

Wait for the **Step 2 - Import File Validation** page to finish. It will reddirect to **Step 3 - Import.**

Make sure that the **Import Option** of **"Delete existing attribute definitions..."** is NOT checked.

Click **Import** and wait for the import to return **Successful.**

The following **System Objects** should now be populated in **Order** and **Site Preferences**

Object Type 'Order' - Attribute Definition Details

On this page you can manage details of your attribute definition. Different options are available depending on the value type of your attribute.
Click **Apply** to create a new attribute definition or save changes to existing attributes.
Some value types support the definition of value lists. In this case you can provide an attribute value id and display name and click **Add** to add new value to the attribute definition.

Select language: | Default | Apply

ID:* vpOrderStatus

Display Name: Virtual Piggy Order Status

Help Text:

Value Type: Enum of String   NOTE: Searchable via query framework.

Mandatory: ☐

Externally Managed: ☐

Select Multiple: ☐

Apply Reset

Object Type 'Order' - Attribute Value Range Definition

This section lists the attribute value definitions of the attribute. Create a new attribute value definition by providing the Value and Display Value in the 'New Value' section below. Press **Apply** to update the attribute value definitions. Click **Reset** to rollback changes. Click **Delete** to delete selected attribute value definitions.

Search Attribute Value Definitions

Value or Display Value: [   ] Find

Select All	Value	Display Value	Default	Sorting
☐	NONE	None	◉	
☐	PENDING_APPROVAL	Pending Approval	○	
☐	AUTHORIZED	Authorized	○	
☐	CAPTURED	Captured	○	
☐	DECLINED	Declined by a Parent	○	
☐	VOIDED	Voided by a Merchant	○	
☐	EXPIRED	Wasn't approved in term	○	
New Value:			○	

Apply Reset Delete

Showing **1 - 7** of **7** items.

<< Back

- Order

  - vpOrderStatus

    - id: vpOrderStatus

    - display name: Virtual Piggy Order Status

    - type: Enum of Strings

    - mandatory: false

    - values:

      - value: NONE, display value: None

      - value: PENDING_APPROVAL, display value: Pending Approval

      - value: AUTHORIZED, display value: Authorized

      - value: CAPTURED, display value: Captured

      - value: DECLINED, display value: Declined by a Parent

      - value: VOIDED, display value: Voided by a Merchant

      - value: EXPIRED, display value: wasn't approved in term

  - vpTransactionId

- id: vpTransactionId

- display name: Virtual Piggy Transaction Id

- type: String

- mandatory: false

- Site Preferences

  - vpApiKey

    - id: vpApiKey

    - display name: API Key

    - type: String

    - mandatory: false

  - vpApiTimeout

    - id: vpApiTimeout

    - display name: API Timeout (ms)

    - type: int

    - mandatory: false

    - min value: 3000

    - default value: 30000

  - vpCurrencyId

    - id: vpCurrencyId

    - display name: Site Currency

    - description: Optional. Please consult your Virtual Piggy contact whether you should put here anything.

    - type: String

    - mandatory: false

    - regex: /^[A-Z]{3}$/

  - vpEndPointURL

    - id: vpEndpointURL

    - display name: Transaction Service URL

    - description: If empty points to a development Virtual Piggy instance

    - type: String

- mandatory: false

- default value:
  https://integration.virtualpiggy.com/Services/TransactionService.svc

o vpManualCapture

- id: vpManualCapture

- display name: Manual Capture

- description: Should be checked if you're using the Manual Capture workflow

- type: Boolean

- mandatory: false

o vpMerchantID

- id: vpMerchantID

- display name: Merchant ID

- type: String

- mandatory: false

- regex: /^[0-9a-f-]+$/

o vpOrderExpirationDays

- id: vpOrderExpirationDays

- display name: Order expires in (days)

- type: Integer

- mandatory: false

- minimum value: 1

- default value: 7

o vpShippingMethodID

- id: vpShippingMethodID

- display name: Shipping Method ID

- description: ID of an active shipping method that will be used for all Virtual Piggy orders. Please ensure that this shipping method is applicable for all locations required.

- type: String

- mandatory: false

- o vpRecvGatewayData

  - id: vpRecvGatewayData

  - display name: Receive Payment Gateway Data (encrypted)

  - description: Enable this preference if you would like to receive Payment Gateway Data (note: enabling this preference requires providing a public key to Virtual Piggy)

  - type: Boolean

  - mandatory: false

- o vpPrivateKey

  - id: vpPrivateKey

  - display name: Private Key

  - description: 256bit encryption Private key used to communicate secured credit card number and CVV. This is only used if Order Management system requires credit card as part of the transaction, or they will reject the order

  - type: String

  - mandatory: false, but is required if vpPaymentGateway data is checked.

- Please follow the following instructions on generating the public and private keys:

  - o Step 1. Get a machine with openssl installed on it (linux/mac have by default, there's a link to a windows binary here: http://www.openssl.org/related/binaries.html)
  - o Step 2. openssl genrsa -out rsaprivatekey.pem 2048
  - o Step 3. openssl rsa -in rsaprivatekey.pem -out publickey.pem -pubout
  - o Step 4. openssl pkcs8 -topk8 -in rsaprivatekey.pem -out privatekey.pem -nocrypt

  - o Provide the public key to OINK, while updating the private key in the local settings.

Virtual Piggy site preferences:

- id: VIRTUAL_PIGGY

- display name: Virtual Piggy Settings

- attributes:

  - o vpEndPointURL

  - o vpMerchantID

  - o vpApiKey

  - o vpApiTimeout

- o vpCurrencyID

- o vpShippingMethodID

- o vpOrderExpirationDays

- o vpManualCapture

- o vpRecvGatewayData

- o vpPrivateKey

System Object Types > Site Preferences - Attribute Groups > Virtual Piggy settings

Object Type 'Site Preferences' - Attribute Definition Assignments

On this page you can assign existing attribute definitions to your attribute group.

Assign Attribute Definition

ID:* [_____] [...] [Add]

Select All	ID	Name	Type	Sortin
☐	**vpEndPointURL**	Transaction Service URL	String	
☐	**vpMerchantID**	Merchant ID	String	
☐	**vpApiKey**	API Key	String	
☐	**vpCurrencyID**	Site Currency	String	⬆ ⬇
☐	**vpShippingMethodID**	Shipping Method ID	String	
☐	**vpOrderExpirationDays**	Order Expires in (days)	Integer	
☐	**vpManualCapture**	Manual Capture	Boolean	
				Unassign

[<< Back]

Set Virtual Piggy Site Preferences for Each Site

For each site that Virtual Piggy used as a payment option, go to Site -> Site Preferences -> Custom Preferences -> Virtual Piggy Settings and set the preferences according to the following screenshot:

Some suggestions:

- Transaction Service URL, Merchant ID, API Key and Site Currency should be requested from Virtual Piggy;
- "Order Expires in" and "Manual Capture" should be set accordingly your Virtual Piggy workflow;
- Shipping Method ID should contain explicit ID of a shipping method configured on your instance for all Virtual Piggy orders (see the screenshot below).

- Please follow the following instructions on generating the public and private keys:

  - Step 1. Get a machine with openssl installed on it (linux/mac have by default, there's a link to a windows binary here: http://www.openssl.org/related/binaries.html)
  - Step 2. openssl genrsa -out rsaprivatekey.pem 2048
  - Step 3. openssl rsa -in rsaprivatekey.pem -out publickey.pem -pubout
  - Step 4. openssl pkcs8 -topk8 -in rsaprivatekey.pem -out privatekey.pem -nocrypt

  - Provide the public key to OINK, while updating the private key in the local settings.

Shipping Methods

Create a Virtual Piggy Payment Method for Each Site

The payment method should be created according to the following screenshot below:

The only required fields are ID, Name and Enabled. The rest of fields can be set according to your requirements.

Provide Callback URLs for Virtual Piggy

To complete the configuration you should provide callback urls for Approval and Reject requests from **Oink** service.

Examples:

https://sandbox1.my.dw.demandware.net/on/demandware.store/Sites-SiteID-Site/default/VPCallback-Approve

https://sandbox1.my.dw.demandware.net/on/demandware.store/Sites-SiteID-Site/default/VPCallback-Decline
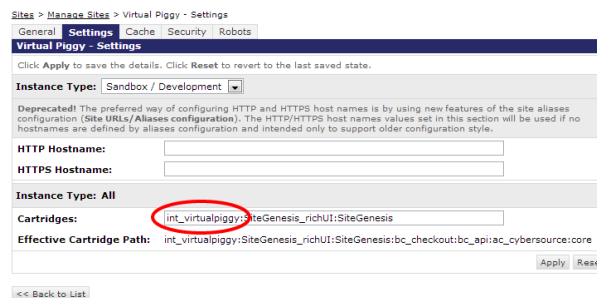
Where:

sandbox1.my.dw.demandware.net – your instance's domain

SiteID – your particular site's identifier

Assign the cartridge to a site

Add the cartridge name into the cartridge list for each site where **Oink** is needed, including Business Manager:

3.3 Custom Code

Integration is based on new templates introduced in SiteGenesis 12.5. Integration points are:

- checkout/pt_checkout_UI.isml
- checkout/cart/pt_cart_UI.isml
- checkout/cart/pt_cart_VARS.isml
- checkout/cart/rightcolumn.isml
- checkout/billing/paymentmethods.isml

1. In **templates/default/checkout/pt_checkout_UI.isml** and **templates/default/checkout/cart/pt_cart_UI.isml** we need to include the Virtual Piggy CSS file. Append this line to the end of both files:

<link rel="stylesheet" href="${URLUtils.staticURL('/css/virtualpiggy.css')}" />

2. In **templates/default/checkout/cart/pt_cart_VARS.isml** we need to include the Virtual Piggy login dialog box. Append this line to the end of the file:

<isinclude template="checkout/cart/vppt_cart_VARS" />

3. In **templates/default/checkout/cart/rightcolumn.isml** we need to include the login button. The following lines should be added just above the "approachingdiscounts" template:

<iscomment>Display button for VirtualPiggy</iscomment>

<isinclude template="virtualpiggy/cart/loginbutton"/>

4. In **templates/default/checkout/billing/paymentmethods.isml** we need to add lines to include VIRTUAL_PIGGY as a payment method. The following lines should be added within the payment-method-options loop:

<iscomment>Ignore VIRTUAL_PIGGY method.</iscomment>

<isif condition="${paymentMethodType.value.equals('VIRTUAL_PIGGY')}"><iscontinue/></isif>

In case your solution is based on an earlier SiteGenesis version or on any other reference application, you have to:

- put contents of *_UI.isml files into <head> section of the pages Cart and Checkout. It can be put it into global header.
- put contents of pt_cart_VARS.isml into bottom of the Cart page (below the **Oink** button include) and any other pages you're going to place the button.

Additionally, the WSDL file may need updated for VirtualPiggy service. To do this, download the latest version from the following address and save it as webreferences/VirtualPiggyService.wsdl:

https://integration.virtualpiggy.com/services/TransactionService.svc?wsdl

In order to add shipping method selection to the VP checkout screen, please refer to the checkout template inside int_virtualpiggy integration in the following location:

Templates/default/checkout/virtualpiggy/checkout.isml.   This file contains typical checkout of the earlier SiteGenesis versions. This line is used to fill in the shipping method selection on VP checkout screen:

```
<div id="shippingmethodform"></div>
```
Followed by shipping method JS code

Payment information is handled in the following pipeline: VirtualPiggy-HandlePayment

Pay special attention to your individual credit card processor, since it will need to be replaced from PayFlow as in the example below.
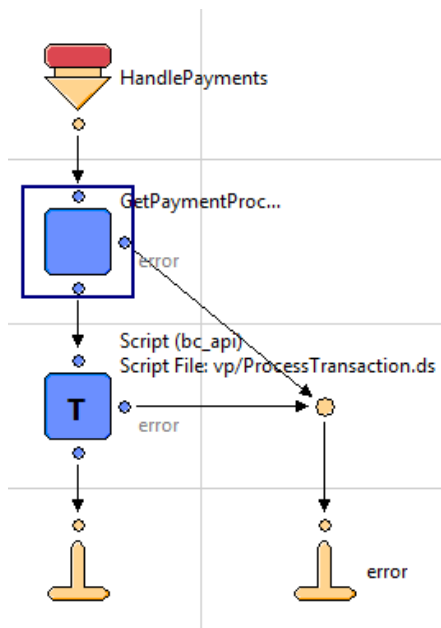
Note: this step is important, if credit card information is communicated back to the storefront.

Vp/processTransaction.ds contains the information regarding fields to be preserved for OMS.

The following fields are now saved into a session and then saved off to the order (as custom attributes described below):

3.4   External Interfaces

The cartridge uses SOAP to interact with the VirtualPiggy service. VirtualPiggy uses http(s) form submit to callback pipelines on the Demandware side to send order status updates.

SOAP requests implemented in the cartridge:

AuthenticateUser	Method to authenticate parent or child and returns a token to use in subsequent calls
CaptureTransactionByIdentifier	For a 2 step transaction process, capture or cancel the transaction
GetAllChildren	Method to return a Parent's list of children that items can be purchased for
GetChildAddress	Method to return Child's address details
GetParentAddress	Method to return a Parent's address details
GetParentChildAddress	Method to return Child's address details

GetPaymentAccounts	Method to return a Parent's payment accounts
PingHeaders	Used to verify connectivity to the web service and verifies validity for the header values vp.MerchantIdentifier and vp.APIkey
ProcessParentTransaction	Method to process parent transaction
ProcessTransaction	Method to process child transaction

The methods are described in detail on Virtual Piggy's Documentation site:
http://docs.virtualpiggy.com/index.php/home/integrate/transaction-services-api

Callback Pipelines

On order status update, the Virtual Piggy service notifies Demandware using callback. There are two callback pipelines that process the following status changes:

- VPCallback-Approve: Processed, Expired, Declined.
- VPCallback-Reject: Reject.

3.5   Testing

There is a dedicated pipeline **VirtualPiggyTest-All** that was developed to test connection and base functions of the integration.

Note: This pipeline is accessed via the browser, however, the pipeline is set to "private" and initially can't be accessed via a browser. To use the pipeline set the start node to "public". When done set the start node back to "private".

It consists of 3 parts:

1. Ping Only.
    a. PingHeaders – only credentials from Business Manager are used for connectivity verification.
2. Child Checkout
    a. AuthenticateUser – authenticate a child using credentials from templates/resources/vptest.properties. Credentials are stored in vptest.data.childlogin and vptest.data.childpassword fields.
    b. GetChildAddress – get a shipping address for the child authenticated.
    c. ProcessTransaction – process a test transaction.
    d. CaptureTransactionByIdentifier – capture the transaction above.
3. Parent Checkout
    a. AuthenticateUser – authenticate a parent using credentials from templates/resources/vptest.properties. Credentials are stored in vptest.data.parentlogin and vptest.data.parentpassword fields.
    b. GetAllChildren – get all children available for the parent and save first from the list for further tests.
    c. GetParentAddress – retrieve the parent's address.
    d. GetParentChildAddress – retrieve the address for the child saved on step b.

       e.   GetPaymentAccounts – retrieve all payment accounts of the parent and store first of them for the transaction below.

       f.   ProcessParentTransaction – process a test transaction for the child saved and using the payment account saved.

       g.   CaptureTransactionByIdentifier – void the transaction above.

4.   Validating data that was set on the order should look similar to this:

Encrypted credit card and card expiration data will be saved as a custom attribute on the order. Also every order will create a credit card transaction with Credit Card information stored in the Payment Transaction. This information can be extracted using regular order export process with private/public data decryption.

4.1 Data Storage

All needed data is stored on Order level in custom fields:

vpCardExpDate	Virtual Piggy Card Expiration	Date
vpCardNumber	Virtual Piggy Card Number (encrypted)	String
vpGatewayAuthCode	Virtual Piggy Payment Gateway Authorization Code	String
vpOrderStatus	Virtual Piggy Order Status	Enum of String
vpTransactionId	Virtual Piggy Transaction Id	String

4.2 Availability

GSI reports 99.9% uptime availability. In case if there is no service, a 404 error will be display by Virtual Piggy. For technical support please contact integration@virtualpiggy.com

4.3 Support

For technical support please contact: integration@virtualpiggy.com

Phone number: 310-853-1950

5.1 Roles, Responsibilities

In case of "Automatic Capture" workflow, no recurring tasks are needed.

In case of "Manual Capture" workflow, a dedicated person is required to make a decision as to whether to proceed with approved orders. Please see section 5.2 for more details.

5.2 Business Manager

In custom Site Preferences a new button appears on the Payment Tab of the Order page.

This button is used to proceed with "Manual Capture" workflow. It works only when Manual Capture is activated in Virtual Piggy Site Preferences and the current order's vpOrderStatus is "Authorized". In this case click on the button will show the following page:

Click on the "Capture" button will send a request to VirtualPiggy to capture the money and will proceed with the order.

Click on the "Void" button will send a request to VirtualPiggy to cancel the transaction and will cancel the order.

5.3 Storefront Functionality

Cart

The cart page will have new button:

Click the button to open a login window with **Oink** Dialog:

Enter login and password credentials in the appropriate fields. They will be sent to **Oink** service to authenticate and to get user type to proceed correctly. After successful authentication, a user is redirected to Parent Checkout or Child Checkout according to his/her user type.

Child Checkout

Child Checkout page contains only ordered product list and a button to proceed.

Clicking on the button will send a request to the **Oink** service. The following responses to the order are:

- **Processed immediately** - order gets approved on **Oink** side and is ready to be processed on Demandware side as an ordinary order
- **Processed as "Approval Pending"** - order is created but waits for an approval from the child's parent
- **Not processed** - order is not processed because of a decision on **Oink** side or an error occurred. If an error occurred, an error message is shown on the page.

Parent Checkout

Parent Checkout page looks similar to the Child Checkout page but it also contains additional controls to:

- choose a child (if multiple exist) that the order is made for
- delivery type (to parent or child address)
- payment method for the order

Please bear in mind that all data except an order itself (e.g. shipping address, payment method etc) are stored and handled on **Oink** side only. Any data saved on a Demandware side is not used.

Order Confirmation

After Parent Checkout and successful Child Checkout (both Processed and Pending Approval) an Order Confirmation page is shown.

6. Known Issues

The web service calls to Virtual Piggy are currently not configured to use any timeout values. Demandware strongly recommends that site preference values be created to configure the timeout value on all web service calls.

7. Release History

Version	Date	Changes
1.2.1	05/31/2014>	New Credit Card capture functionality, so it can be added to the payment gateway for OMS processing. This process is PCI compliant and utilizes regular DW payment gateway logic.
1.2.0	12/13/2013>	Virtual Piggy rebranded to "Oink"
1.1.0	10/17/2012	Initial release