Gateway Ticketing Systems

Generic Web Pay Plugin Interface Specification

Table of Contents

1 History

Date	Revision	Version of Galaxy	Who
12/13/2016	Basic documentation of Web Pay plugin API	7.0+	MTW
2/1/2017	Add Chapter 5	7.0+	BUS

2 Introduction

The core eGalaxy Web Store supports external payment processors, with the ability to redirect customers to their websites to pay for their tickets. The interfaces in the GTS.Plugin.PluginContracts assembly can be implemented by eGalaxy Web Store end-users to define custom payment processor plugins. The supplied SDK includes all documentation and sample code you should need to implement your own payment processor that plugs into eGalaxy Web Store's payment subsystem.

2.1 SDK contents

Path	File / Path	Description
Bin\		
	GTS.Plugin.PluginContracts.dll	.NET assembly that defines the interfaces used in the payment processor subsystem
Doc\		
	WebPayPluginAPI.pdf	This document
Sample\		
	C#\	Sample payment processor plugin developed using Visual Studio 2015 and .NET 4.5.

3 Interfaces

3.1 *IAbstractPluginModule*

eGalaxy Web Store provides this type definition for external payment processors. Developers implement this interface and plug it into eGalaxy Web Store to create their own payment processors. This interface defines the basic structure of the custom configuration form fields that will be injected into the Web Admin page to fill in basic account information for a payment processor.

3.1.1 Properties

Name	Type	Description
ModuleType	string	The name of the plugin module. It should be unique across all plugins. This is a read-only property.

3.1.2 Methods

3.1.2.1 IEnumerable<IPluginConfigField> GetConfigFields();

This is called by eGalaxy Web Store to retrieve the custom configuration fields defined by the plugin.

Parameters: None

Return value: An `IEnumerable` providing field definitions (`IPluginConfigField`) for each custom configuration field defined by the plugin.

3.1.2.2 bool ValidateConfigField(string key, string value, out string errorMessage);

This is called by eGalaxy Web Store to validate the data entered in the custom fields.

Parameters:

key	The name of the custom field, as provided by the IEnumerable returned from the `GetConfigFields()` call.
value	The text that the administrator has entered into the field.
errorMessage	If the administrator's input is invalid, this should be filled in with a message explaining the problem.

Return value: True if the administrator's input is valid, false otherwise. If the result is false, the `errorMessage` parameter should have a value assigned to it explaining why it didn't validate.

3.2 IIndirectPluginModule

This interface extends `IAbstractPluginModule`, adding a method for submitting a payment.

3.2.1 Properties

This interface does not define any properties.

3.2.2 Methods

3.2.2.1 void StartPayment(IGenericPayment payment, HttpResponse httpResponse);

This is called by eGalaxy Web Store to submit a payment to the custom payment processor.

Parameters:

payment	An `IGenericPayment` instance containing information about the payment to be submitted
response	A `System.Web.HttpResponse` object for the plugin to write the redirect page to.

This method is not expected to submit the payment itself, but rather to generate an HTML page which, when displayed in the customer's browser, will redirect the browser to the payment processor's site, with the appropriate details filled in, and allow them to process the payment.

3.2.2.2 bool ValidatePayment(IGenericPayment payment, HttpRequest, out string errorMessage);

This is called by eGalaxy Web Store to allow the custom payment processor to validate the response to a payment sent by the `StartPayment` method. This method allows your payment provider to verify that the incoming payment confirmation works as expected.

Parameters:

payment	An `IGenericPayment` instance containing information about the payment to be submitted
request	A `System.Web.HttpRequest` object containing a callback to the `IGenericPayment.AcceptUrl` URL.
errorMessage	If the validation fails, this should be filled in with a message explaining the problem.

Return value: `true` if the payment is valid, or `false` if it fails for any reason. When returning `false`, the `errorMessage` parameter should be filled in with an explanation of the validation problem that occurred.

3.3 IGenericPayment

This interface defines the data in a request to be sent to a payment processor. This interface is implemented by the eGalaxy Web Store and is sent to the plugin through the `IIndirectPluginModule.StartPayment`() method.

3.3.1 Properties

Name	Type	Description
FopCode – should Fop be capitalized?	Integer	The FOP (Form Of Payment) code for this payment, corresponding to the `FOPs.FOPCode` values in the database.
OrderID	String	The OrderID number generated for this order by eGalaxy Web Store.
OrderAmount	Decimal – how many decimal places?	The total amount of the payment for this order.
AcceptUrl	String	The URL that the payment provider should return the user to after a successful payment
CancelUrl	String	The URL that the payment provider should return the user to if the user cancels or does not successfully pay
SubmitType	String	This property tells the plugin whether this is a real transaction or a test. Real transactions will have a `SubmitMode` value of “Production,” whereas for test transactions it will be “Test”.
ConfigFields	IEnumerable<IPaymentConfigField>	A sequence of `IPaymentConfigField` instances describing the custom field data provided by the user.
Order	String	Order XML returned as string containing all the details about the order

3.3.2 Methods

There are no methods on this interface

3.4 IPaymentConfigField

This interface defines the data filled in by the administrator, in a request to be sent to a payment processor. This interface is implemented by the eGalaxy Web Store and is sent to the plugin through the `IIndirectPluginModule.StartPayment`() method, through the `IGenericPayment.ConfigFields` property.

3.4.1 Properties

Name	Type	Description
key	String	The name of the custom field, as provided by the dictionary returned from the `GetConfigFields()` call.
value	String	The data that the administrator has entered into the custom field.

3.4.2 Methods

There are no methods on this interface

3.5 IPluginConfigField

This interface defines the properties of custom fields to be displayed for the user to fill in. Objects implementing this interface are provided to the eGalaxy Web Store by the plugin, as part of the response to the `IAbstractPluginModule.GetConfigFields()` method.

3.5.1 Properties

Name	Type	Description
Key	String	The name of the custom field.
Label	String	The text of the bold field label that will be written to the configuration page.
Description	String	The text of the field description that will be written to the configuration page below the Label
Required	Bool	If this is `true`, the configuration page will treat this custom field as a required value. (i.e., It's invalid for the administrator to leave it blank.)
RequiredValidationMessage	String	If `Required` is `true`, this property gives the error message that will be displayed if the field is left blank.
EncryptionEnabled	Bool	This property is used for password fields. If it's `true`, the field will be displayed in HTML as a password-style field (user input will be masked instead of displayed in clear text) and its value will be encrypted when stored to the database.

3.5.2 Methods

There are no methods on this interface

4 Creating a payment processor plugin

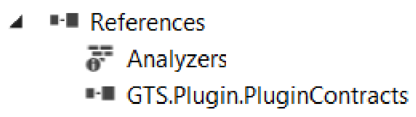
Gateway Ticketing Systems has created its own plugin implementations for the eGalaxy Web Store using C#. The following is a guide for using those tools.

4.1 C# in Visual Studio

The idea when creating a payment processor plugin using C# is that you create an ordinary .NET assembly, containing a class that implements the plugin interface. We've included a sample plugin built with Visual Studio 2015 and .NET 4.5. See the SDK zip file for the entire solution. You should be able to use the provided source, compile it, plug it right into eGalaxy Web Store and watch it work. You can start with our provided solution and just modify it to your needs, or start fresh. If you start fresh, there are a few things you'll need to do before firing up Visual Studio.

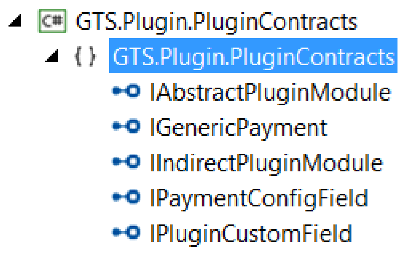
4.1.1 Add a reference to the DLL

Create your solution or project in Visual Studio. Next, you need to add a reference to the assembly containing the eGalaxy Web Store interfaces (GTS.Plugin.PluginContracts.dll).


References
Analyzers
GTS.Plugin.PluginContracts

When you add the reference, you browse to the DLL and add it to the project.

After you're done adding the reference, you can double-click on the reference to view it in the object browser. It should look something like this:


GTS.Plugin.PluginContracts
GTS.Plugin.PluginContracts
IAbstractPluginModule
IGenericPayment
IIndirectPluginModule
IPaymentConfigField
IPluginCustomField

4.1.2 Write some code

Next you develop your implementation of the `IIndirectPluginModule` interface. See the file DemoPlugin.cs in the SDK for sample code.

There are a few things to point out that could help your effort:

- The class definition resembles the following:

  ```
  [Export(typeof(IAbstractPluginModule))]
  [PartCreationPolicy(CreationPolicy.NonShared)]
  [ExportMetadata("DisplayName", "Demo Plugin")]
  [ExportMetadata("Description", "This is demo plugin to test MEF ")]
  [ExportMetadata("Version", "2.5.1")]
  [ExportMetadata("GuidNumber", "0a9c0850-7c84-476f-8ff6-3437a9428f70")]
  public class DemoPlugin : IIndirectPluginModule
  ```

  NOTE: The `Export` attribute is important. If you leave it out, eGalaxy Web Store will not be able to locate your plugin. The `ExportMetadata` attributes provide metadata about your plugin, and the “DisplayName” and “Version” metadata properties are particularly important: they’re used by eGalaxy Web Store to look up your plugin by name and version. Also the GuidNumber ExportMetadata value must be a unique GUID value.

- Next, simply implement the `GetConfigFields()`, `ValidateConfigFields()`, `StartPayment()` and `ValidatePayment()` methods and the `ModuleType` property, and the plugin will be ready to test.

5 Adding a payment processor plugin to eGalaxy Web Store

There are a few steps you'll need to take to get a payment processor up and running in the eGalaxy Web Store. The following is a high level guide.

5.1 Localization Strings

Within the SDK\Sample folder is a SQL script file named “LocalizationScripts.sql.” This file contains the 2 localization strings that are used by the eGalaxy Web Store to display the name of your Plugin.

```
EXEC GTSAddWSLocalizationRow @GroupID, 'CheckoutPage/Index', 'PaymentATypeTitle', 'Payment Type Title', 0, 0
EXEC GTSAddWSLocalizationRow @GroupID, 'CheckoutPage/Index', 'PaymentATypeDescription', 'Payment Type Description', 0, 0
```

The localization script will need to modified to match your plugin definition. Specifically, the text PaymentAType will need to be replaced with the `ModuleType` value from your plugin. So if your ModuleType is NewPaymentPlugin, the two lines would be:

```
EXEC GTSAddWSLocalizationRow @GroupID, 'CheckoutPage/Index', 'NewPaymentPlugin Title', 'Payment Type Title', 0, 0
EXEC GTSAddWSLocalizationRow @GroupID, 'CheckoutPage/Index', 'NewPaymentPluginDescription', 'Payment Type Description', 0, 0
```

Once the script has been modified, provide it to a Galaxy administrator so that it can be run against their Galaxy (not web) database. Once run, they can refresh the localization strings within their merchant(s) following the normal process that they would use when receiving new localization strings via the GalaxyDatabase.sql file.

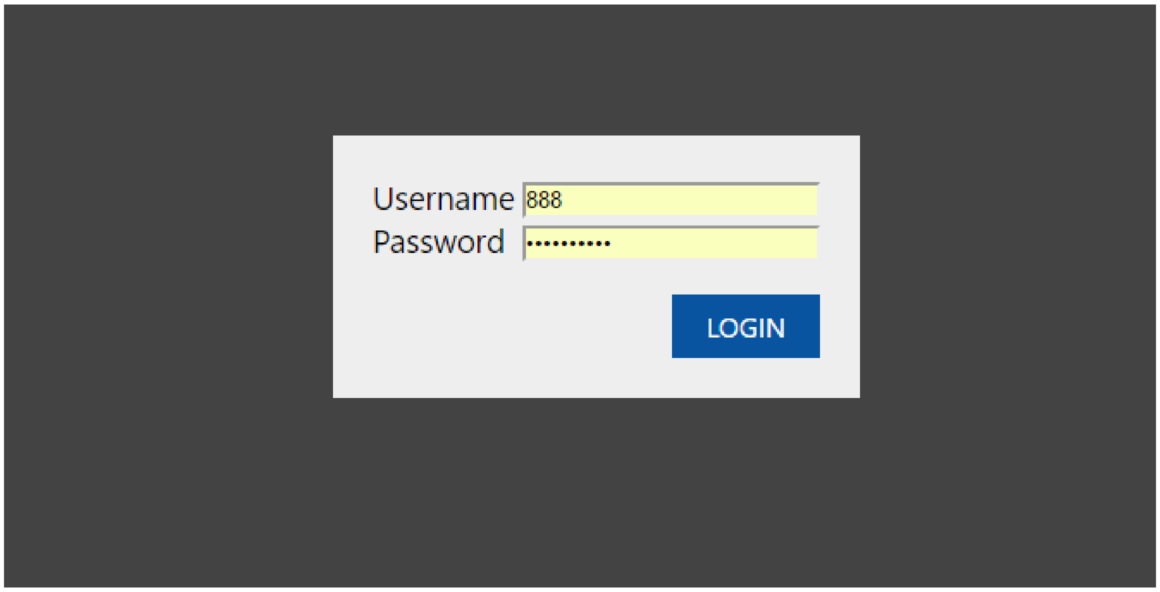
5.2 Installation

For Gateway hosted Web stores, please contact your Gateway representative to have your plugin loaded to your Web store.

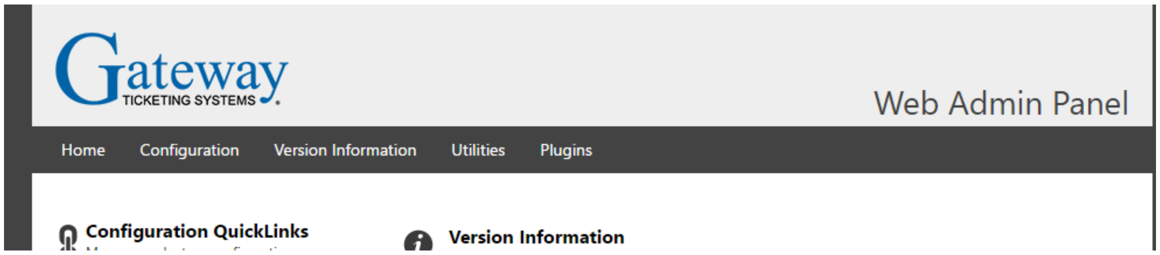
For self-hosted Web stores, the plugin DLL must be included in these folders:

- WebAdmin\Plugins
- WebStore\Plugins

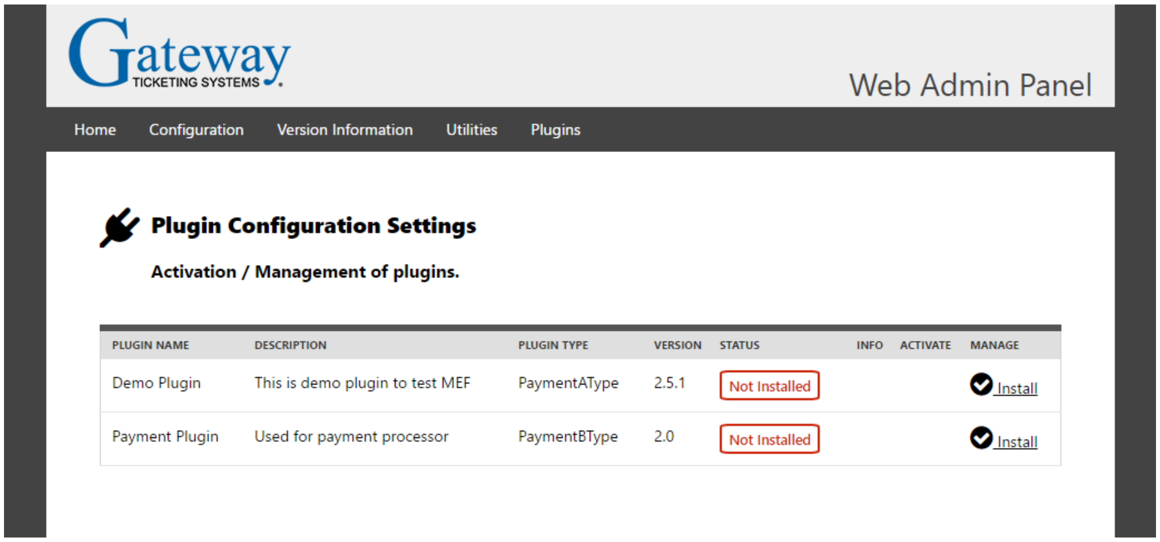
Step 1: Once the plugin dll has been added to the “Plugins” folder within “Web Admin” and “Webstore” hosted on the server, log into the Web Admin panel using your credentials.



Step 2: Navigate to the “Plugins” tab on the top menu section of the Web Admin panel.



Step 3: In the plugins panel, you can view all the available plugins created implementing `IAbstractPluginModule` that are added in the “Plugins” folder. The “Plugin Name”, “Description“ and “Plugin Type” columns display your unique plugin name, a short description of your plugin and unique plugin type respectively that are added as part of the metadata of the plugin.

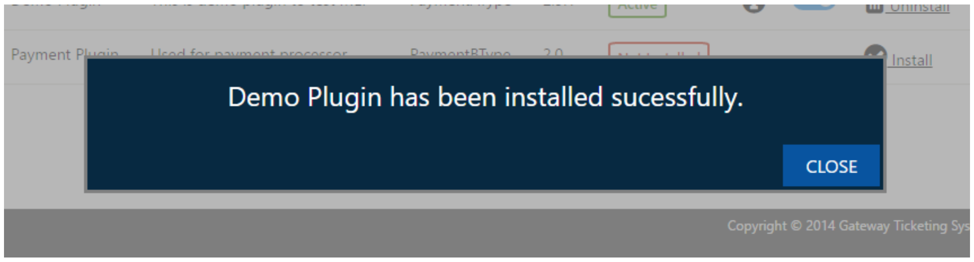


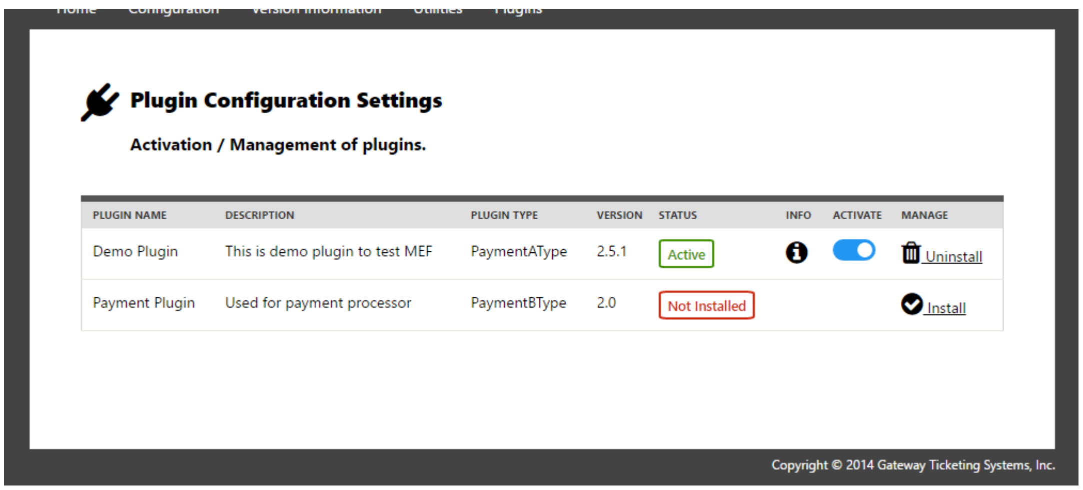
Plugin Configuration Settings

Activation / Management of plugins.

PLUGIN NAME	DESCRIPTION	PLUGIN TYPE	VERSION	STATUS	INFO	ACTIVATE	MANAGE
Demo Plugin	This is demo plugin to test MEF	PaymentAType	2.5.1	Not Installed			Install
Payment Plugin	Used for payment processor	PaymentBType	2.0	Not Installed			Install

Step 4: Install the plugin by clicking the “Install” link for each plugin row. The plugin installation message will be displayed after it has been successfully installed and added to the Installed Plugin list. Also, the “STATUS” column should display as “Active” instead of “Not Installed”.

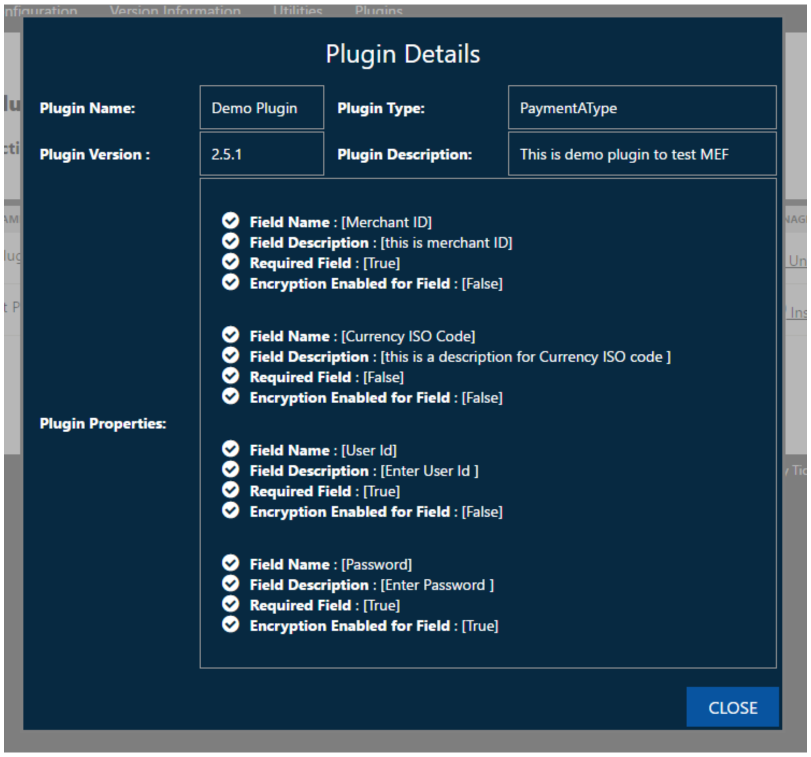

Demo Plugin has been installed sucessfully.


Plugin Configuration Settings

Activation / Management of plugins.

PLUGIN NAME	DESCRIPTION	PLUGIN TYPE	VERSION	STATUS	INFO	ACTIVATE	MANAGE
Demo Plugin	This is demo plugin to test MEF	PaymentAType	2.5.1	Active	●	●	Uninstall
Payment Plugin	Used for payment processor	PaymentBType	2.0	Not Installed			Install

Step 5: Once the plugin has been installed, you can view the details about the plugin by clicking on info ℹ


Plugin Details

Plugin Name:	Demo Plugin	**Plugin Type:**	PaymentAType
Plugin Version :	2.5.1	**Plugin Description:**	This is demo plugin to test MEF

Plugin Properties:

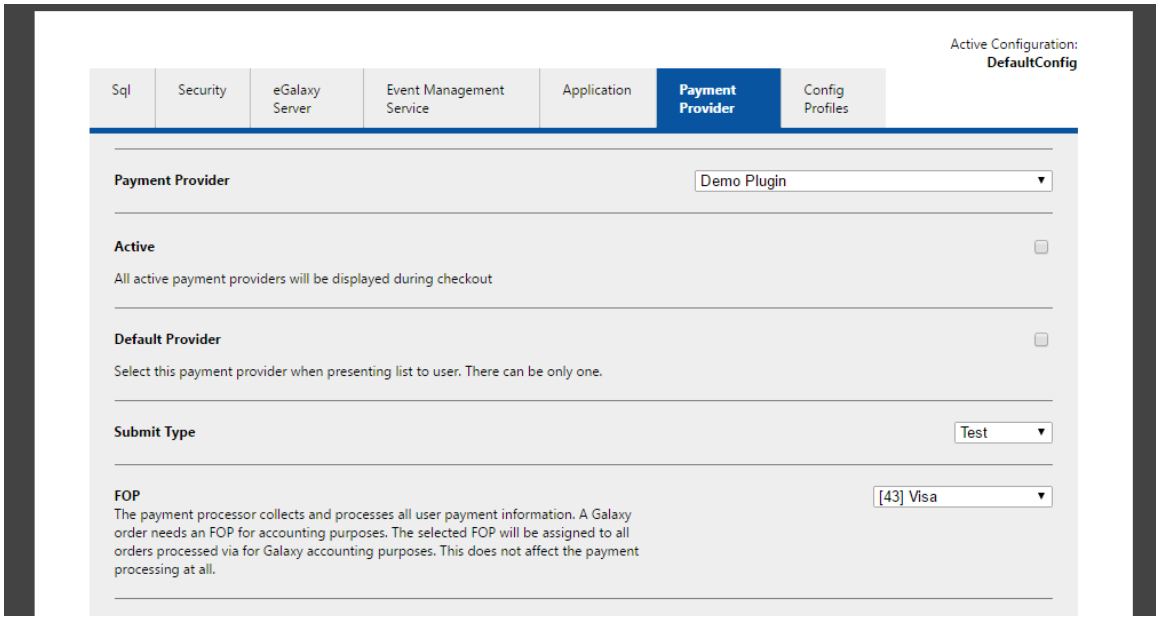
- ✓ **Field Name :** [Merchant ID]
- ✓ **Field Description :** [this is merchant ID]
- ✓ **Required Field :** [True]
- ✓ **Encryption Enabled for Field :** [False]

- ✓ **Field Name :** [Currency ISO Code]
- ✓ **Field Description :** [this is a description for Currency ISO code]
- ✓ **Required Field :** [False]
- ✓ **Encryption Enabled for Field :** [False]

- ✓ **Field Name :** [User Id]
- ✓ **Field Description :** [Enter User Id]
- ✓ **Required Field :** [True]
- ✓ **Encryption Enabled for Field :** [False]

- ✓ **Field Name :** [Password]
- ✓ **Field Description :** [Enter Password]
- ✓ **Required Field :** [True]
- ✓ **Encryption Enabled for Field :** [True]

Step 6: Once the plugin has been installed, it is now available for selection to enable the payment processor provided through the newly installed plugin by navigating to the “Payment Provider” tab and selecting it in the payment provider section dropdown list. All of the configuration fields required by the plugin are displayed once the new payment plugin has been selected.



Active Configuration: **DefaultConfig**

Payment Provider — Demo Plugin

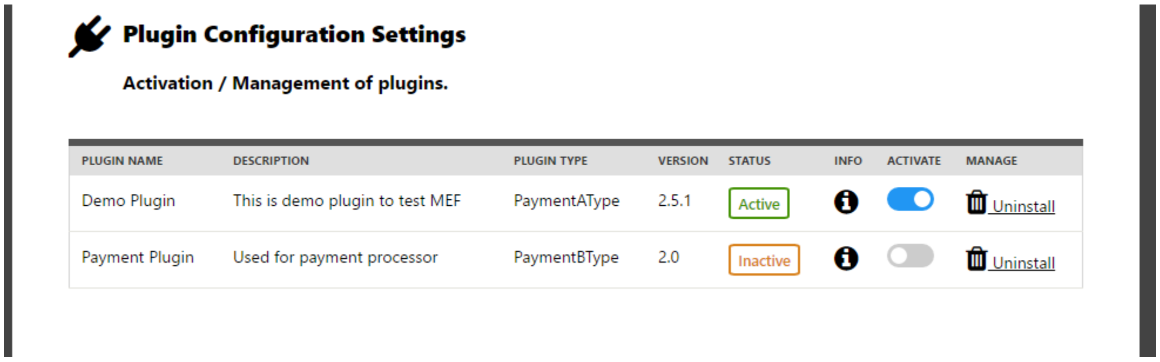
Active ☐
All active payment providers will be displayed during checkout

Default Provider ☐
Select this payment provider when presenting list to user. There can be only one.

Submit Type — Test

FOP — [43] Visa
The payment processor collects and processes all user payment information. A Galaxy order needs an FOP for accounting purposes. The selected FOP will be assigned to all orders processed via for Galaxy accounting purposes. This does not affect the payment processing at all.

Step 7: The installed plugin can also be deactivated or uninstalled on the “Plugins” panel.



Plugin Configuration Settings

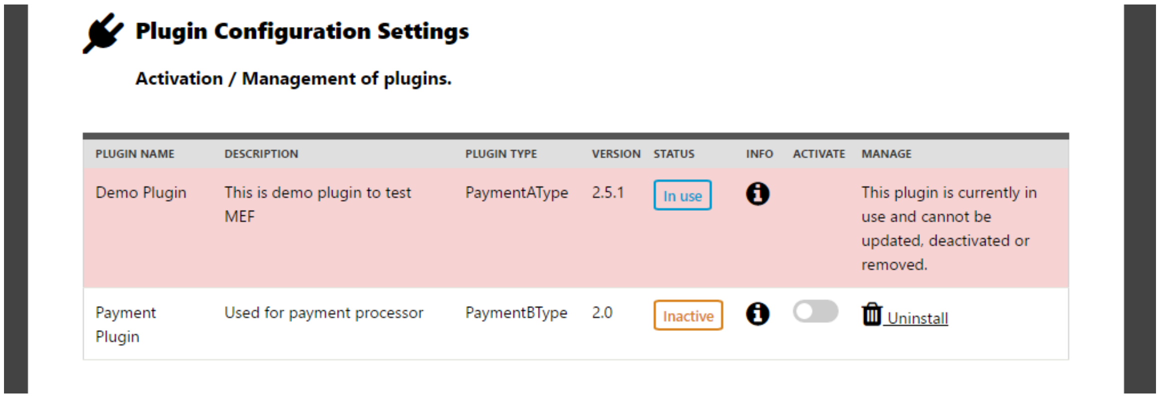
Activation / Management of plugins.

PLUGIN NAME	DESCRIPTION	PLUGIN TYPE	VERSION	STATUS	INFO	ACTIVATE	MANAGE
Demo Plugin	This is demo plugin to test MEF	PaymentAType	2.5.1	Active			Uninstall
Payment Plugin	Used for payment processor	PaymentBType	2.0	Inactive			Uninstall

When you manage a plugin, the following options are available:

Option	Description
Status	This ready-only column displays the state of each plugin: ✓ **Not Installed** - The DLL file exists in the Plugins directory, but the plugin has not been installed on the Web Store ✓ **Inactive** - The plugin is installed, but it does not appear in the Payment Provider list. ✓ **Active** - The plugin is installed on the Web Store and can be configured through the Configuration > Payment Provider panel. ✓ **In use** - The plugin is set as an active payment provider. It cannot be updated, deactivated, or removed.
Info	Click the icon to view a read-only summary of the plugin's properties.
Activate	Click the slider button to toggle the plugin's status between active and inactive.
Manage	Click the link to install or uninstall the plugin.

Note: If the plugins are currently being used or selected as the default payment provider in the Configuration >> “Payment Provider” tab of the Web Admin panel, those plugins cannot be deactivated or uninstalled. The “STATUS” of those plugins are displayed as “In use”. Also, both the “de-activate” toggle button as well as the “Uninstall” link are hidden. To deactivate or uninstall the plugin, the plugin has to be removed from the default payment provider.



Plugin Configuration Settings

Activation / Management of plugins.

PLUGIN NAME	DESCRIPTION	PLUGIN TYPE	VERSION	STATUS	INFO	ACTIVATE	MANAGE
Demo Plugin	This is demo plugin to test MEF	PaymentAType	2.5.1	In use			This plugin is currently in use and cannot be updated, deactivated or removed.
Payment Plugin	Used for payment processor	PaymentBType	2.0	Inactive			Uninstall