Create Payment Provider

# Create a New Provider

Open Visual Studio and create a new project, using the “RE\_PaymentProvider” project template.

Rename the project and place the project in “..\DesktopModules\DNNrocketModules\”.

## System Data Interface

You need to create a system data interface for your provider, so RocketEcommerce knows it exists. The interface contains setup configuration data.

Open the “RocketSystem” System Data page. (../desktopmodules/dnnrocket/adminsystem.html)

On systems list, enter “rocketecommerce”.

Copy the “manualpay” interface to the same system “Rocket E-commerce”. (Go out and back in the system details to view the interface).

Rename the data to match the new name of the new payment provider and save.

**Key:** This will be used in the code to identify the interface.

**Admin Namespace:** The class that will be called for administration of setting.

**Provider Namespace:** The class that will be called for the payment gateway.

**Assembly:** The Assembly Name.

**Default Command:** Default command that will be called.

**Control Relative Path:** The folder path used to get the provider templates.

## Rename Project

Rename the project name and the solution name in Visual Studio.

Rename ALL strings of “payprovider” to the new provider identifier in the system data.

\*\* IMPORTANT \*\* USE MATCH CASE.

At this point you can compile the project. By default, the paybox template is used as a base for the payment provider. This code will need to be adapted to the new gateway.

## Add to Portal Payment Provider List

The new payment provider will need to be added to the list of portal payment gateways.

Go into Portal List. Select the portal. Go into the Payment providers tab. Check the new provider and save.

Activate the new payment provider on the website

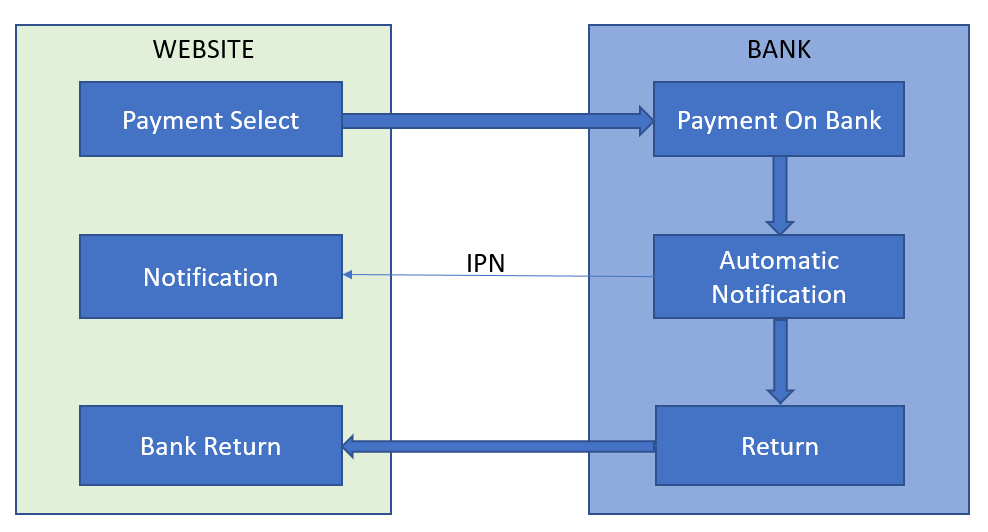
Go into “Settings>Payment Methods”. (Any error might be linked to the system interface data)

Select the new Payment Provider.

Clicking activate, will activate the payment provider in the website.

# How it Works?

This payment provider method uses the traditions “redirect method”. When the user clicked the “Pay” button, a html page is created on the server with all the information required. This is then posted by replacing the browser document and the user is redirected to the bank, where they can pay and then return to the website after payment.



We usually only code for what the website is doing. The bank side is out of our control. However, we often need to pass settings to the bank. These settings are sometimes in the admin of the bank system and don’t need to be passed.

# IPN

The IPN (Instant Payment Notification) is dealt with by a API url. The data returned by the IPN can be in different formats, depending on the bank. However, with always need to return the IPN to a standard place in Rocket E-commerce.

“<Your Engine Domain>/Desktopmodules/dnnrocket/api/rocket/action”

This should also have some parameters added to it, so the action API knows what provider to run. We must tell it the systemkey (s): “rocketecommerce”, the command (cmd): “rocketecommerce\_notify”, the interface to use (interfacekey): “rocketecommerce” and the payment provider interface (p): “paypal”.

Because the systemkey and the interfacekey are the same as the command, we only need to pass the command and payment provider key in the URL.

“<YourEngineDomain>/Desktopmodules/dnnrocket/api/rocket/action?cmd=rockecommerce\_notify&p=paypal”

# Return

The return is a simple redirect back to the website, with the paymentkey as a parameter in the URL. “…?key=XknQu30bN0eZVfLKIjxUQ”. The call to the API will be made on page display and update the payment record as required.