|  |
| --- |
| CloudCommerce Factory |
| Getting Started |
| Installation and deployment of the CCF solution |

|  |
| --- |
| CloudCommerce Factory |

# I Prerequisite

## Compilation

The following elements are required to be able to compile the solution:

* Windows SDK and .NET framework 4.0 (included in the Install folder, file winsdk\_web.exe): this will install msbuild as well as some needed dlls
* Visual Studio (for Web): provides web development dlls needed by the solution. Note that the full version of Visual Studio contains those as well
* PostSharp 2.1 (version 2.1.7.35): an open source third party application

This element may already be installed on the server / computer in which case you won’t have to install them.

Regarding PostSharp, you may install it before launching the first compilation. In the other case, a window will pop-up, allowing you to register the product, via *Register a license -> Register PostSharp Starter Edition.*

## Web server configuration

You will need the following to run the web sites:

* IIS, which can be installed via the “Add windows feature” menu in control panel (choose the standard installation by ticking the box)
* IIS module rewrite rule (installer can be found in Install folder, file urlrewrite2.exe)
* Register the ASPNET framework in IIS, via the sell in administrator mode:
  + On Windows 8 and Server 2012 or higher: dism /online /enable-feature /all /featurename:IIS-ASPNET45
  + On Windows 7 and Server 2008: C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet\_regiis.exe –ir

## SQL Server

The websites use SQL Server database, so you need SQL Server 2008 or higher, with Advanced Services.

You can use the Express version, which are available at the address below, selecting the executable SQLEXPRADV\_x64\_ENU (64bits OS) or SQLEXPRADV\_x86\_ENU (32bits OS):

* <http://www.microsoft.com/en-us/download/details.aspx?id=29062>

During the installation, make sure you select the Mixed Authentication Mode (*Database Engine Configuration > Server Configuration > Authentication Mode: Mixed Mode*).

# II Compilation

The script MPfPackage.bat compiles all the projects of the solution, in the output directory DeliveryBuild, and a zip version of each of theme is created in DeliveryPackage, for deployment purpose.

You can also open the sln files with Visual Studio and compile the projects as you would do for any other C# project.

In Visual Studio, the project that you need to set as starting project is MPf.Web.Administration, MPf.Web.Front or MPf.Web.API depending of the sln file.

To compile the service project (included in MPf.API.sln), you need to do it manually by right-clicking and building the MPf.Service.Job.Windows project. The output executable is located in MPf.Service.Job.Windows\Debug\bin and should be run as administrator.

# III Deployment

## Database

Two SQL scripts manage the creation of the database needed by the solution. They are located in the Install folder.

They must be used by sqlcmd, via Powershell or the Windows shell.

Depending on your installation of SQL Server, you might need to specify the server name by adding -S SERVER\_NAME\INSTANCE\_NAME to the sqlcmd command lines below.

The first script only create the database with the wanted name:

> sqlcmd -i .\00.createDatabase.sql -v dbName="DATABASE\_NAME" [-S SERVER\_NAME\INSTANCE\_NAME]

The second step is to create a login on the server and a user for this login on the database newly created. The syntax is as follow:

> sqlcmd -i .\01.createLoginAndUser.sql -v loginName="LOGIN\_NAME" password="PASSWORD" dbName="DATABASE\_NAME" userName="USER\_NAME" [-S SERVER\_NAME\INSTANCE\_NAME]

Next, you have to add the .NET application specific permission to the database, with the following command:

C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet\_regsql.exe -E -A all -d DATABASE\_NAME -S SERVER\_NAME

In case of trouble, you can also run the executable alone to open a graphical interface.

## Configuration

The deployment on various environments is managed by the properties files: they are basic configuration file, containing key/value pair.

During deployment, the script takes the environment name as parameter and find the appropriated file, based on its name (i.e. dev.properties for an environment called dev). The final file config is then built with the value of this file applied to the empty \*\_.config file.

Basically, for the front office website for example:

Web\_.config + dev.properties = Web.config for “dev” environment

The 4 parts of the solution (back office, front office, API and services) have their own folder in the DeliveryPackage root folder. In each of these folders, a conf subfolder contains the properties files (only template.properties at the moment, the example file).

The first step is to copy the template.properties file in the same folder and rename it local.properties. Then, open it and and edit the values.

Most of the keys are self-explanatory, here is a short explanation on some of the most important:

* **application:** the application name, used to build the log file name
* **mpf\_connectionstring:** contains the server name, Initial Catalog is the name of the database, User ID is the LOGIN\_NAME used previously in SQL Server, PASSWORD too
* **website\_url:** url of the images website you will be creating later (described below)
* **resources\_absolute\_path:** absolute path to the images website folder (also described below)
* **templates\_absolute\_path:** absolute path to the front office website folder
* **front\_url:** url of the front office website
* **web.front.theme:** allow you to change theme
* smtp section (**mail.\***): config of the smtp server used to send mails
* **batch.path**: absolute path to the folder where the batch files will be kept (must be equal in all properties file)
* **log.file.path**: absolute path to the folder where the log files will be created
* **system\_webServer\_rewrite\_canonical\_rule:** regex form of the front url, ie. ^front\.localhost\.com$ : special characters escaped with \, ^ at the start and $ at the end
* **system\_webServer\_rewrite\_canonical\_redirect:** completeurl of the front with /{R:1} at the end (ie. http://front.localhost.com/{R:1}

Once done, the script CreateConfigFileLocal.ps1 located in the root folder creates the config files in each of the projects, based on the local.properties files.

The **web.front.theme** variable must correspond to the name of the folder containing your views (in MPf.Web.Front/Themes). The default theme we provide is called motoshop.

## IIS Setup

We describe in this section the how to setup IIS using the scripts we provide. Nevertheless, keep in mind that you can configure it by yourself.

The setup is done using Powershell scripts (ps1) that are preferably run in Administrator mode.

By default, execution of powershell scripts is not allowed. In case you get an error message regarding “Execution\_Policies”, run the following command in Powershell:

Set-ExecutionPolicy unrestricted

The 3 websites all need to first be created in IIS using the script 00.SetUpIIS.ps1, located in the DeliveryPackage folder.

The following parameters must be given:

* **siteName:** name of the site
* **physicalPath:** absolute path to the folder that will contain the website
* **userName:** identity under which the website will be installed
* **userDomain:** domain of the user
* **userPassword:** password of the user
* **url:** url of the website (do not include protocol, ie front.localhost.com, without http://)

Others (optional and less important) parameters exist:

* **port:** port to bind to (80 by default)
* **scheme:** protocol (http by default)
* **ip:** specific ip to bind to (all by default)
* **identityType:** type of identity (SpecificUser by default)
* **logPath:** IIS log file path
* **logFormat:** IIS log format (standard W3C by default)
* **applicationPoolName:** equals to siteName by default, will be created if non existent

Usage example:

>./00.SetUpIIS.ps1 –siteName “front” –physicalPath “D:\sites\” –userName “bill” –userDomain “bill-PC” –userPassword “\*\*\*\*\*\*” -url “front.localhost.com”

Regarding the **physicalPath:** if you are compiling the solution with Visual Studio, you can set the path to the folder containing the csproj of the project (ie. path\to\solution\MPf.Web.Administration or MPf.Web.Front or MPf.Web.API). It makes it easier to modify the code and test it quickly, without having to run any other scripts.

You also need to create a website to host the pictures generated by the solutions (mostly the products’ pictures). The syntax is the same as for the others websites, using the same script (00.SetUpIIS.ps1). The **url** and **physical path** you provide for this site are respectively the **website\_url** and **resources\_absolute\_path** of the properties you create earlier.

## Deployment

The deployment script 01.InstallSite.ps1 is used for websites deployment. It takes as parameter the project to deploy (backoffice, frontoffice or api) and automatically finds the zip files in DeliveryPackage folders.

The whole zip file will be extracted to the physical path of the given site, then the Web.config will be created by merging the Web\_.config and the properties file.

The script has two mandatory parameters:

* **project:** backoffice, frontoffice or api
* **env:** the environment, this parameter must correspond to a properties file name

If only those parameters are given, a list of available websites will be displayed and the user will be prompted to choose. You can pass one of these parameters to search for the website and avoid this step:

* **siteName:** as it appears in IIS
* **url:** binding of the website

Note that this script will by default copy the current content of the selected website to a backup path, by default backup folder is created in the current path. You can specify a more convenient backup folder thanks to the **backupPath** parameter.

Additionally, some parameters allow you to modify the default behavior:

* **zipFile:** path to the zip file containing the website
* **configurationFile:** path to the properties file
* **skipBackup**

Usage example:

>./01.InstallSite.ps1 –project frontoffice –env “CI-2” -siteName “front” –backupPath “D:\backup\”

Or to deploy without backing up:

>./01.InstallSite.ps1 –project frontoffice –env “CI-2” -siteName “front” –skipBackup

The script 02.InstallService.ps1 install and deploy the service of product import.

Note that the user has to be different from the websites user.

The parameters are mostly the same as for other projects:

* **env:** the environment, this parameter must correspond to a properties file name
* **physicalPath:** absolute path to the folder that will contains the service
* **userName:** identity under which the service will be installed
* **userDomain:** domain of the user
* **userPassword:** password of the user

As for the websites, a backup is done at each deployment. You can specify its location thanks to the **backupPath** parameters.

The parameters **zipFile** and **configurationFile** allow you to specify the location of the zip and the configuration file.

You can also specify the **serviceName** and **serviceDisplayName** parameters(MPf.Service.Job.Windows by default)**.**

Usage example:

>./02.InstallService.ps1 –env “CI-2” -physicalPath “D:\services\” –userName “service-user” –userDomain “bill-PC” –userPassword “\*\*\*\*\*\*” –backupPath “D:\backup\”

## Activation

Once you are done setting up the website, it’s time to activate your solution: the process is pretty simple, you just have to launch the administration back office website. You will be prompted for you license key.

You can then log in to your back office with the following default information:

* login: admin
* password: ^(#3"r=f<9j8GWN

After the first login, we strongly recommend you to change the default password of the administrator and the operator. This can be done in the “Users” menu of the administration back office, submenu “Users”. You will see the two users and by clicking on each of them you will be able to change their password.