# Building the custom Extensions for OPA

## About Custom Functions

Any custom function is simply an extension of the CustomFunction class provided by the determinations-engine.

Custom Functions are extensions are added to an OPA project by adding an extensions folder in the Development directory of the OPA project. Then each library of extensions you want to provide is put in a folder (here we have the TextFunctions).

**Extensions**

|

+--**TextFunctions**

| |

| +--extension.xml

| |

| +--**Lib**

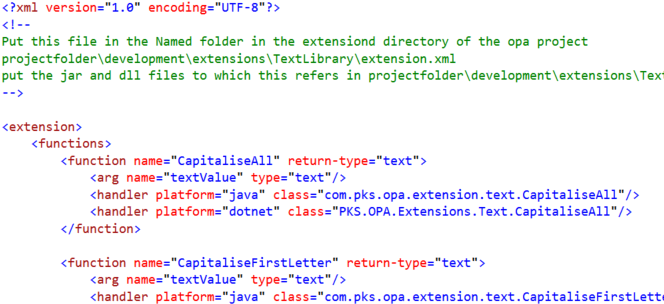
| |

| +--JavaStrings.jar

| +--DotNEtStrings.dll

Every OPA extension library must contain an extensions.xml to define the extensions to be linked and the values to be passed. There is then a jar and a dll in the LIB folder to provide the Java and .net versions of the extension classes.

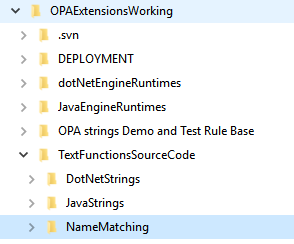
There can be multiple custom function classes in each JAR or DLL file and multiple JAR and DLL files too, OPA will iterae them for the required functions automatically. We just need to reference the fully qualified class names in the XML. Here is a small snippet showing a CapitaliseAll() function that returns a text value and takes a text argument. You can see both Java and .NET classes referenced.



You need to create both Java and .NET versions of every function because the OPA Debugger uses .NET and the test environment uses Apache Java server, the deployment could be either a Java WAR file or a .NET Zip file.

## Build tools and source location

The source is stored in a folder called “OPAExtensionsWorking” it has the following folders:



## .NET

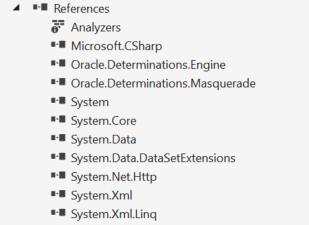
Source Code in DotNetStrings folder

Complied using Visual Studio, an SLN file is present in the source folder

### References.

Nothing particularly exciting here, really just need to add the two Oracle.Determinations references and these may be found in the provided dotNetEngineRuntimes folder. The rest were just added by the VS environment

It is recommended to avoid adding anything other than core assemblies because the .net dependency in OPA is quite early version. For example the latest regular expression classes did not operate correctly.



Some tests are included using the VS test framework.

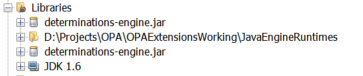
## Java

Source code is in the JavaStrings folder. The original build was using netbeans 8.2 In netbeans you may simply open the folder as a project.

You require JDK 1.6 (SE only not EE), this is because later java may not operate properly with either the debugger or deployment environment.

### Dependencies

* JDK 1.6 (copy provided because Oracle may stop providing at some point)
* Determinations-engine.jar (you will find this in the provided JavaEngineRuntimes folder.



If using another compiler, there is nothing exciting about the build, just find the java source files and add them to the project.

Some test files are included using JUNIT

## OPA test folder

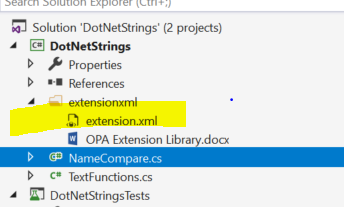
There is a simple OPA rule base included containing a relationship and a few attributes. This is intended for debugging purposes.

In general the processs is to

1. build the initaial .NET and Java versions to a reasonable degree,
2. copy the Extensions folder into the OPA project
3. then interact with OPA through the OPA debugger, updating the .NET version until all is well
4. return to the Java version and update to match the .NET
5. In OPA, export the debugging session as a test case then check that the java version operates properly by running the regression tester.

## Deployment

Ensure you have updated the word document that specifies the usage of the extensions and the XML file!



Deployment involves copying the XML, JAR, DLL and (and the word document that describes the usage) into the correct places in the Extensions folder.

There is a DEPLOYMENT folder, and within it a batch file that will carry out all the copying to create a new extensions folder that can be simply copied as required into the OPA development folder.

So

1. Ensure the XML and docx files are up to date
2. Build Java version
3. Build .net version
4. Go to deployment folder
5. Click on the REFRESH\_DEPLOY.BAT file
6. When done (couple of seconds) copy the extension folder to wherever needed.