Expressions – Using OGNL

Overview

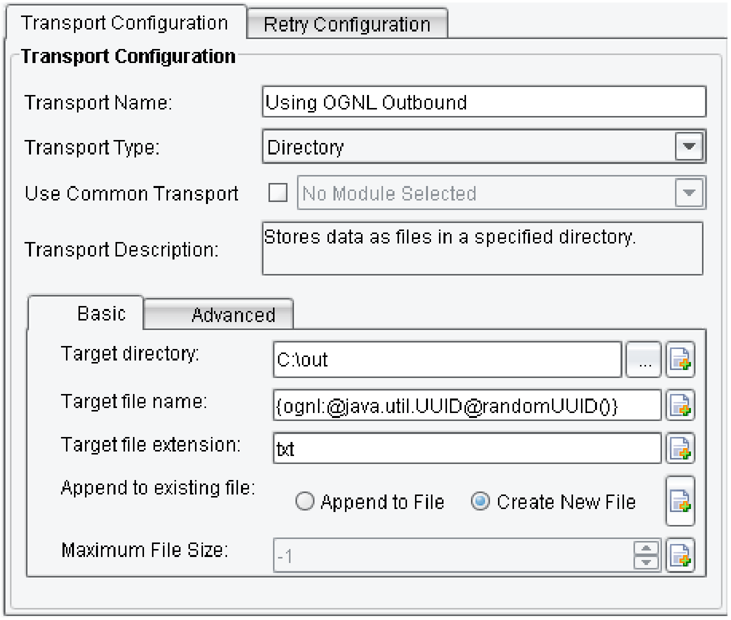
In this lab we will cover using OGNL expressions in configuration item values to allow for runtime evaluation of programmatic instructions. This lab expands on concepts covered in “Expressions – Properties,” so users are expected to be familiar with that content.

Steps

1. Define a Directory Transport

   Suppose we need a unique ID for a file's name. Using the java class “java.util.UUID” and it’s static method called “randomUUID()” (which, as the name implies, generates a new “UUID” with a semi-random, unique value) create the file name.

   That's the entirety of our expression. Placed into the Directory Transport configuration, it looks like this:



Transport Configuration

Transport Name: Using OGNL Outbound

Transport Type: Directory

Use Common Transport ☐ No Module Selected

Transport Description: Stores data as files in a specified directory.

Target directory: C:\out

Target file name: {ognl:@java.util.UUID@randomUUID()}

Target file extension: txt

Append to existing file: ○ Append to File ● Create New File

Maximum File Size: -1

Whenever this Transport now executes, we'll get randomly named files, such as “264a5d08-55fe-44a9-9f7f-6121bee1fa50.txt”

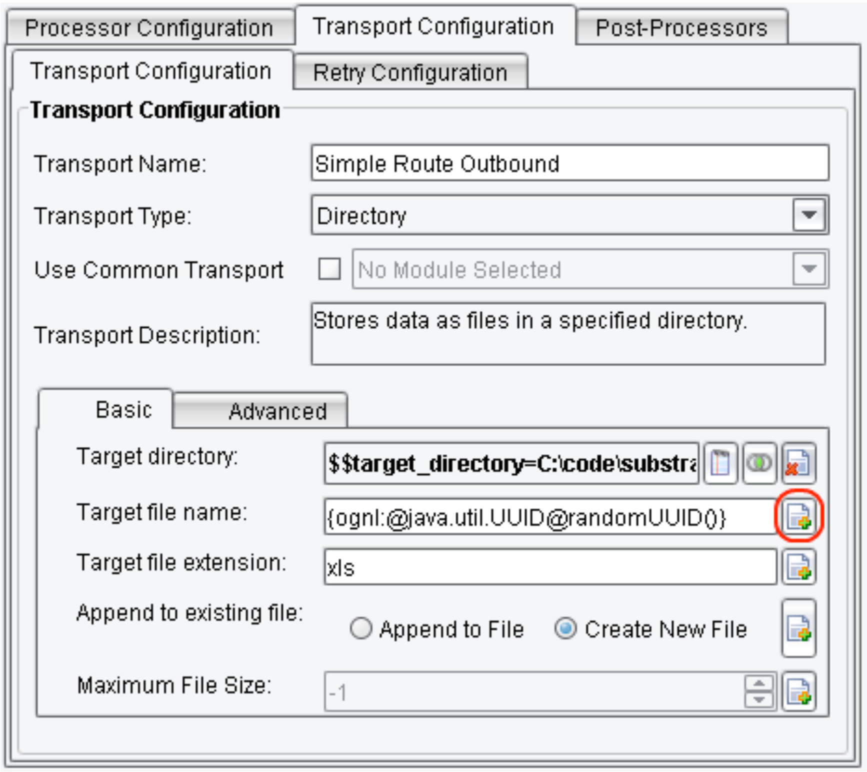
One other concept we'll briefly note is the OGNL context. The default context used by the eiConsole is the “TransactionAttributes” class, giving you access to transaction attributes. You can use this by specifying “#method” to access some method for the current transaction's attributes.

```
{ognl:<method>}
```

To fetch the attribute “com.pilotfish.FileName,” which is created by the Directory Listener when files are picked up, we would provide:

```
{ognl:getAttribute(“com.pilotfish.FileName”)}
```

We can activate the Enhanced Expression buttons by clicking the button on the right of our field:



Transport Configuration

Transport Name: Simple Route Outbound

Transport Type: Directory

Use Common Transport □ No Module Selected

Transport Description: Stores data as files in a specified directory.

Target directory: $$target_directory=C:\code\substra

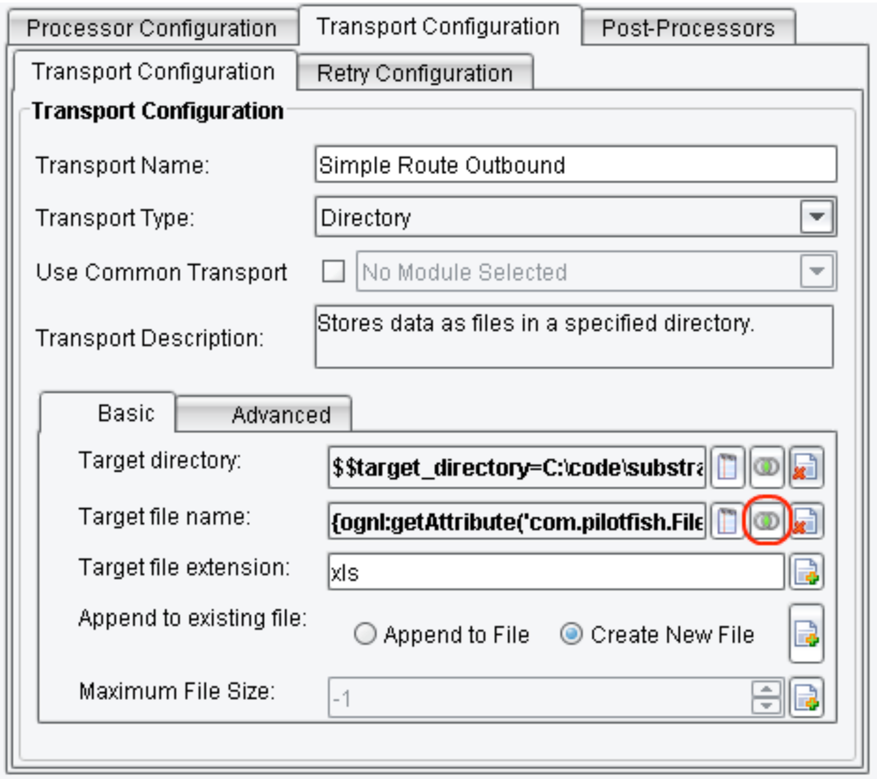
Target file name: {ognl:@java.util.UUID@randomUUID()}

Target file extension: xls

Append to existing file: ○ Append to File ● Create New File

Maximum File Size: -1

We can then invoke the OGNL builder by clicking the middle button:



Transport Configuration

Transport Name: Simple Route Outbound

Transport Type: Directory

Use Common Transport □ No Module Selected

Transport Description: Stores data as files in a specified directory.

Target directory: $$target_directory=C:\code\substra

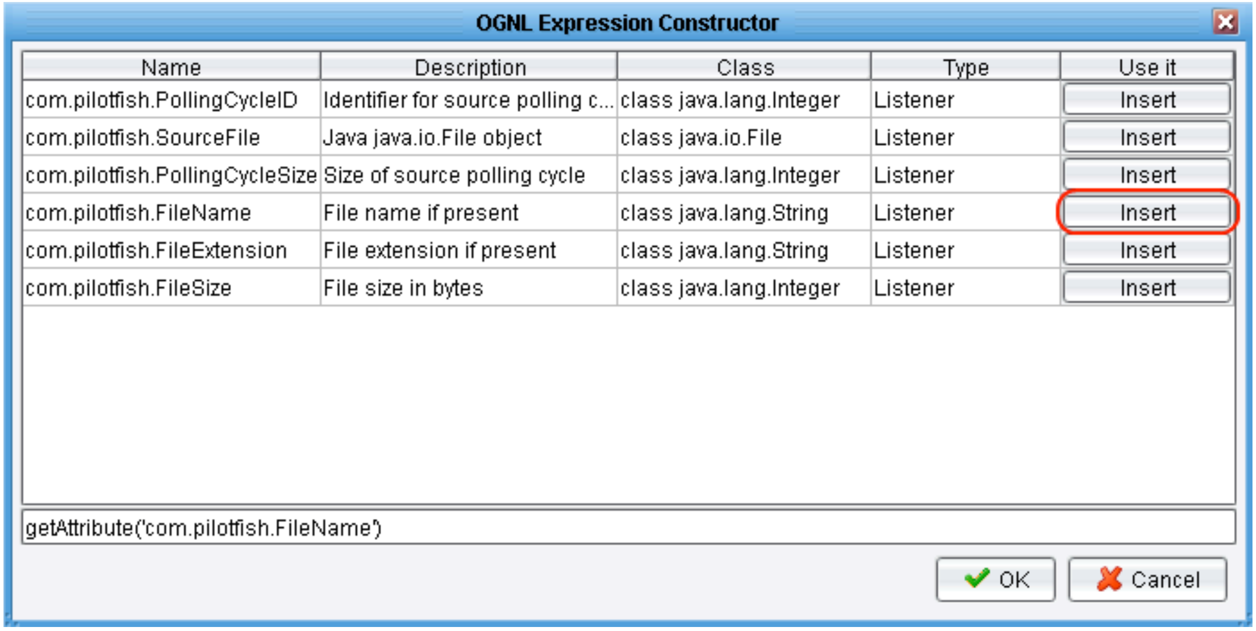
Target file name: {ognl:getAttribute('com.pilotfish.File

Target file extension: xls

Append to existing file: ○ Append to File ● Create New File

Maximum File Size: -1

Last step is to insert one of the Registered Attributes created by our Directory Listener:



OGNL Expression Constructor

Name	Description	Class	Type	Use it
com.pilotfish.PollingCycleID	Identifier for source polling c...	class java.lang.Integer	Listener	Insert
com.pilotfish.SourceFile	Java java.io.File object	class java.io.File	Listener	Insert
com.pilotfish.PollingCycleSize	Size of source polling cycle	class java.lang.Integer	Listener	Insert
com.pilotfish.FileName	File name if present	class java.lang.String	Listener	Insert
com.pilotfish.FileExtension	File extension if present	class java.lang.String	Listener	Insert
com.pilotfish.FileSize	File size in bytes	class java.lang.Integer	Listener	Insert

getAttribute('com.pilotfish.FileName')

Note that you can edit the OGNL expression in bottom field.

You can also read environment properties from OGNL using the following syntax:

```
@com.pilotfish.eip.config.EnvironmentConfigReader@getValue("X")
```

To instantiate an object from OGNL using the following syntax:

```
(new java.text.SimpleDateFormat('yyyyMMdd_HHmmss')).format(new
java.util.Date())
```

Reference: http://commons.apache.org/proper/commons-ognl/language-guide.html