

# Develop FileNet P8 BPM 4.0 custom components using Eclipse

Create a custom workflow for business process management

Skill Level: Introductory

[Dao-Quynh L. Dang \(quynhdang@us.ibm.com\)](mailto:quynhdang@us.ibm.com)  
Senior Software Developer  
IBM

[Indrajit Poddar \(ipoddar@us.ibm.com\)](mailto:ipoddar@us.ibm.com)  
Senior Software Engineer  
IBM

31 Dec 2008

The IBM® FileNet P8 4.0 product suite includes the Business Process Manager (BPM). This product provides the ability for users to define custom action in a workflow using the Java programming language. For example, the user can add sending email, logging work progress for a particular participant, setting document properties, and querying external databases. This article presents steps for developing such custom actions using Eclipse.

## Develop P8 4.0 BPM custom components using Eclipse

### Prerequisites

It is assumed that you are knowledgeable with the different components of the IBM FileNet P8 platform, such as Application Engine (AE), Content Engine (CE), and Process Engine (PE). A working P8 environment with the latest P8 4.0 fix packs applied is assumed, and a minimum P8PE4.0.2-001 is required. In addition, familiarity with Eclipse Integrated development environment (IDE) is preferred. However, step-by-step explanations are presented.

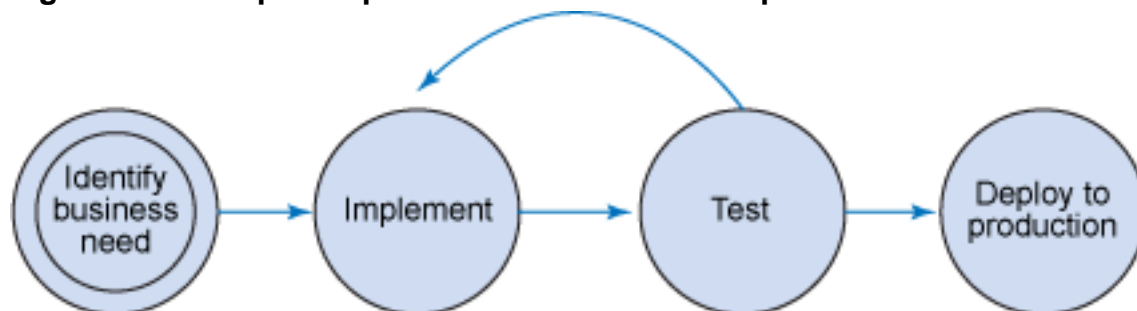
## Introduction

Custom actions in a workflow are operations of a component queue. For example, provided in the standard installation of the P8 BPM server is the CE\_Operations component queue. This queue provides operations such as file, unfile, getStringProperty, setStringProperty. A Java component queue has the operations defined from the public methods of the Java class.

Figure 1 illustrates the following development process for a custom component:

1. Identify the business need for the component
2. Implement the custom component
3. Iterative test and debug the custom component in a P8 BPM development system
4. Deploy the component to the P8 BPM production system

**Figure 1. Development process for a custom component**



Being able to develop and debug the custom component in the BPM development environment in Eclipse is important to guarantee the correctness of the component and expedite the production deployment of the component.

This article shows you the following:

- How to set up Eclipse to be a P8 development environment using the pre-installed components for P8 BPM.
- Steps for develop, deploy, and debug a simple Hello World component without leaving the Eclipse IDE. Tips and tools are also described to help simplify the test-debug iteration process. This includes:
  - The HelloWorld class
  - The shortcut to configure the Java class to be a BPM Java component queue in Eclipse, instead of through the Process Configuration Console application.

- The shortcut to start the P8 BPM Component Manager in Eclipse, and thus enable the debugging of the Java component.

## P8 development environment

A P8 application uses the Content Engine (CE) API and the Process Engine (PE) API through the following libraries:

- The CE API: `Jace.jar` and the transport protocol for the API. For a stand-alone application, the recommended transport protocol is the CE API through the Web services interface. The library can be found on the PE server, `\fnsw\CE_API` directory and its subdirectories.
- The PE API: `pe.jar`, `pe3pt.jar`, `peResources.jar` — These files can be found on the PE server, in the `\fnsw\bin` directory.

### Set up Eclipse

See the [Resources](#) section for downloading Eclipse 3.3 or later. In this article, an Eclipse project is created, containing references to the CE and PE API libraries. Follow these steps to create the user libraries in Eclipse:

1. Create a directory for the Eclipse workspace, for example, `c:\SampleWS`.
2. Create directories for the P8 CE and PE libraries under this workspace directory, for example:
  - `C:\SampleWS\P8Libraries\CE_API`
  - `C:\SampleWS\P8Libraries\PE_API`
3. Copy the entire directory tree content from the PE server `\fnsw\CE_API` to `C:\SampleWS\P8Libraries\CE_API`.
4. Copy the **pe.jar**, **pe3pt.jar**, and **peResources** from the PE server `\fnsw\bin` directory to the corresponding `PE_API`, `c:\SampleWS\P8Libraries\PE_API`.
5. Open Eclipse, and select the **C:\SampleWS** directory for the Workspace directory.
6. Create **PE\_API** and **CE\_API** user libraries for use in the Eclipse workspace by selecting the **Window > Preferences** menu option in Eclipse.

- **PE\_API user library** contains

```
C:\SampleWS \P8Libraries\PE_API \pe.jar  
C:\SampleWS \P8Libraries\PE_API\ pe3pt.jar  
C:\SampleWS \P8Libraries\PE_API\peResources.jar
```

- **CE\_API user library** contains

```
C:\SampleWS\P8Libraries\CE_API\lib\Jace.jar,  
C:\SampleWS\P8Libraries\CE_API\lib\log4j.jar,  
C:\SampleWS\P8Libraries\CE_API\wsi\lib\wasp.jar
```

These libraries can then be added to the Java projects created in the workspace to access the PE and CE APIs.

### Command line arguments for P8 applications

The command line to start a Java application using the P8 4.0 PE API needs to have the following classpath and JVM parameters:

- classpath pe.jar;wasp.jar;Jace.jar;pe3pt.jar;peResources.jar
- Djava.security.auth.login.config=[CE\_API]\config\jaas.conf.WSI
- Dwasp.location=[CE\_API]\wsi
- Dfilenet.pe.bootstrap.ceuri=[CE URI for WSI]

In this article, the different applications share similar JVM configurations. After one configuration is set up, it can be duplicated and revised for other classes.

## A Hello World Java component

A Hello World component queue will be developed to be used in a workflow. Associated with this component queue is the HelloWorld Java class. The class contains one method, sayHello, with the following method signature

```
public String sayHello(String customer);
```

While this seems to be a simple method, the objective of this section is to show shortcuts enabling the component development entirely in Eclipse. Therefore, simplifying the iterative process of developing, testing, and debugging custom components for P8 BPM.

This section describes:

- The HelloWorld class
- How to configure the Java class to be a BPM Java component queue in Eclipse, instead of through the Process Configuration Console application.
- How to run the P8 BPM Component Manager in Eclipse. Thus, enabling the debugging of the Java component.

## Develop the component

To develop the component, perform the following steps:

1. Create a Java project named **PEComponents**.
2. Add the user libraries **CE\_API** and **PE\_API** to the project.
3. Create a new package, **com.filenet.pe.components**.
4. Create a new class for the package HelloWorld.
5. Edit the class to implement the sayHello method as follows:

```
package com.filenet.pe.components;

public class HelloWorld {
    public String sayHello (String customer)
    {
        return "HELLO "+customer+". How was your day on "+
            new java.util.Date().toString();
    }
}
```

6. Build the project.

## Configure the component queue in Eclipse

### JAAS in a custom component

JAAS is the authentication mechanism used by BPM and its custom components. Typically, a JAAS configuration file (defined through the system property `java.security.auth.login.config`) contains stanzas of JAAS contexts. A JAAS context provides the login modules participating in the authentication process. For example, a Java component may need to authenticate with a database for its operations. In this case, the login module handling the DB authentication should be defined for the JAAS context of the component.

At the minimum, a JAAS context should contain the `filenet.vw.server.VWLoginModule`. This means that the provided user information should exist for both the directory server used by the P8 and the custom component.

Normally, the class needs to be packaged into a JAR file to be referenced by the Process Configuration Console applet when the component queue is created. This manual step can be time consuming and error-prone. This section shows how to deploy the component directly within Eclipse.

To run a P8 BPM application, the following information should be gathered:

- The CE server URI
- The PE user and password
- The PE connection point

To configure a component queue, the following information should be gathered:

- The component queue name
- The name of the Java Authentication and Authorization Service (JAAS) context for use by the component queue
- The user and password to be used in the JAAS context for the component
- The name of the Java class
- The methods to be used as operations for the queue

Starting in P8PE4.0.2-001, the programmatic creation of the component queue is possible with the tool `filenet.vw.integrator.base.PEComponentQueueHelper`. Table 1 lists the parameters for the tool.

**Table 1. Parameters for `filenet.vw.integrator.base.PEComponentQueueHelper`**

<b><code>/PEuser PEUser</code></b>	PE User
<b><code>/PEpw PEPw</code></b>	PE Password
<b><code>/PErouter PEConnectionPointName</code></b>	PE Connection Point
<b><code>/queue componentQueueName</code></b>	Name of the component queue to be created
<b><code>/JAAS JAASContext</code></b>	JAAS Context
<b><code>/user JAASUsername</code></b>	JAAS user name
<b><code>/pw JAASPassword</code></b>	JAAS password
<b><code>/class componentClassName</code></b>	The class name for the component queue.
<b><code>/methods name1,name2,....</code></b>	<b>Optional:</b> Names of the methods of the classes to be imported as the component queue

operations. If not specified, all the public methods of the class are imported as operations for the queue.

### Set up Eclipse to run `filenet.vw.integrator.base.PEComponentQueueHelper`

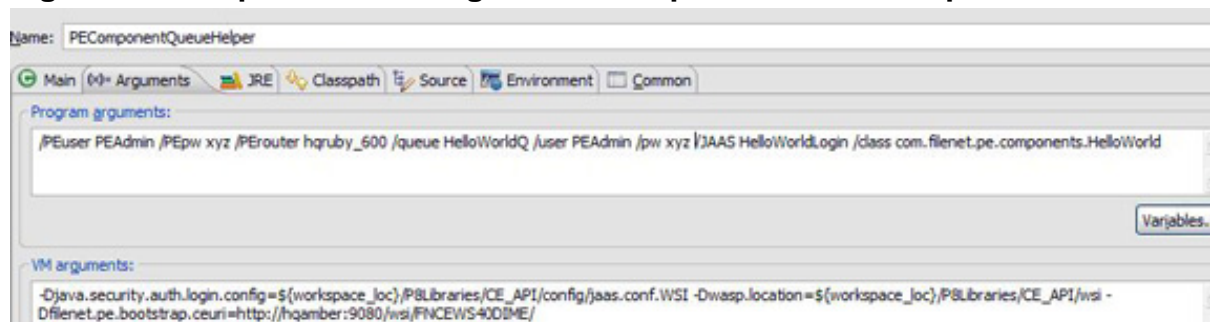
To set up Eclipse to run `filenet.vw.integrator.base.PEComponentQueueHelper`, perform the following steps:

- Create a Run Configuration for `filenet.vw.integrator.base.PEComponentQueueHelper`. Select the **Run/Open Run Dialog...** menu option.
- Double-click on the Java application to create a new configuration.
- Enter **PEComponentQueueHelper** for Name and **`filenet.vw.integrator.base.PEComponentQueueHelper`** for the Main class.
- On the Arguments tab, set the following values using the information in Table 2. Replace the parameter values with that of your environment. For an example, see [Figure 2](#).

**Table 2. Parameters for the `filenet.vw.integrator.base.PEComponentQueueHelper`**

<b>Program arguments</b>	<code>/PEuser PEUser /PEpw PEPasssword /PErouter PEConnectionPoint /queue HelloWorldQ /user PEUser /pw PEPasssword /JAAS HelloWorldLogin /class com.filenet.pe.components.HelloWorld</code>
<b>VM Arguments</b>	<code>-Djava.security.auth.login.config=\${workspace_loc}/P8Libraries/CE -Dwasp.location=\${workspace_loc}/P8Libraries/CE_API/wsi -Dfilenet.pe.bootstrap.ceuri=http://CEServer:CEPort/wsi/FNCEWS4</code>

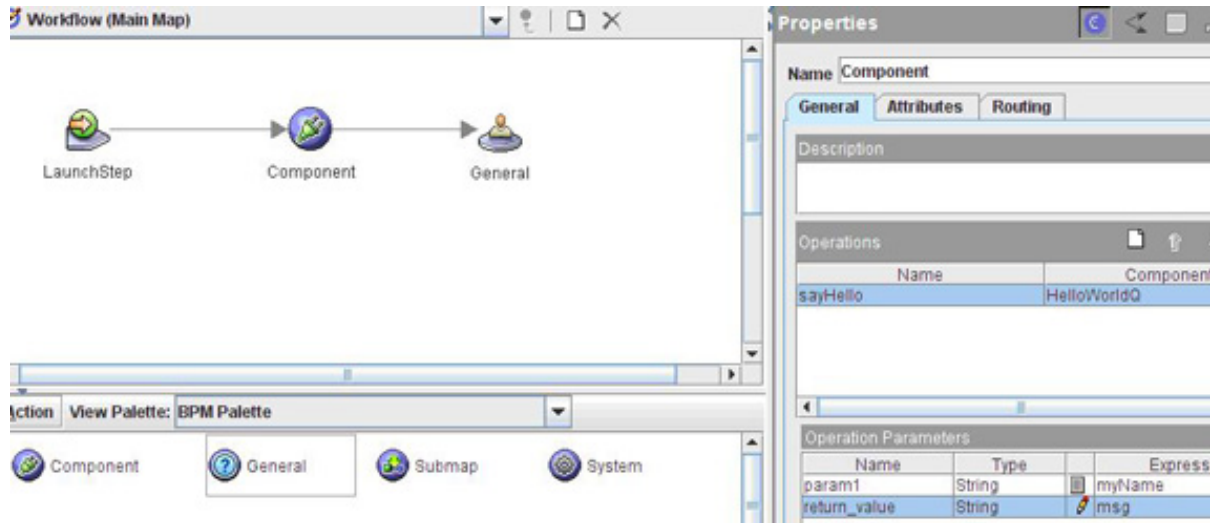
**Figure 2. Example of the configuration tool parameters in Eclipse**



After running the `PEComponentQueueHelper`, use the Process Configuration Console to verify that the component queue has been created with the specified queue name, JAAS information, and operations.

Use the Process Designer applet to create and launch a workflow with a step using the component queue. See Figure 3 for an example.

**Figure 3. Sample workflow to use custom components**



**Run Component Manager in Eclipse**

**Component Manager command line parameters**

To examine the command line starting Component Manager on the Application Engine server, perform the following steps:

1. Exit the Process Task Manager on the application engine.
2. Add the following line to <AE Installed Dir>\Router\taskman.properties  
**filenet.ae.taskman.cm.advanced=true**
3. Restart Process Task Manager.

The Java Security Tab of a Component Manager instance now has the Show command button. Clicking on this button displays the Component Manager command line.

As a tip, if a custom component needs additional parameters, the parameters can be passed through additional system properties (-D JVM option) to the Component Manager.

Component Manager runs as an independent Java application on the application engine. It is a PE API application polling component queue for work. It can be set up to just process specific queues instead of all component queues. At initialization, Component Manager instantiates an object of the class for the custom component and sets up the necessary JAAS framework that fit the component needs for accessing its own resources (such as a JDBC connection).

When a work object arrives in the queue, the Component Manager invokes the corresponding method of the class on the class object. Multi-threaded custom components run in multiple threads, where each thread has its own class object. Debugging a custom component really means debugging Component Manager.

This section discusses how to set up Eclipse to run Component Manager, which allows for an easy way to debug the component.

## Component Manager JVM options and parameters explained

A sample of the command line to execute an instance of Component Manager is given below:

### Listing 1. Sample of command line to execute an instance of Component Manager

```
"C:/Program Files/FileNet/AE/Router/JRE/bin/java"
-Dfilenet.pe.bootstrap.ceuri=http://hqamber:9080/wsi/FNCEWS40DIME/
-Dwasp.location=C:/Program Files/FileNet/AE/Router/./CE_API/wsi"
"-Djava.security.auth.login.config=
  C:/Program Files/FileNet/AE/Router/taskman.login.config"
-Djava.security.policy=C:/Program Files/FileNet/AE/Router/taskman.policy"
-Dfilenet.PE.NOWS=true -Xrs
-cp "C:\Program Files\FileNet\AE\Router\lib\pe.jar;C:\Program Files\FileNet\AE\Router\
lib\peResources.jar;C:\Program Files\FileNet\AE\Router\lib\xml-apis.jar;C:\Program Files\
FileNet\AE\Router\lib\xercesImpl.jar;C:\Program Files\FileNet\AE\CE_API\lib\Jace.jar;
C:\Program Files\FileNet\AE\Router\lib\saaj.jar;C:\Program Files\FileNet\AE\Router\lib\
wsdl4j.jar;C:\Program Files\FileNet\AE\CE_API\wsi\lib\wasp.jar;C:\Program Files\FileNet\
AE\Router\lib\mailapi.jar;C:\Program Files\FileNet\AE\Router\lib\axis.jar;
C:\Program Files\FileNet\AE\Router\lib\axis-schema.jar;C:\Program Files\FileNet\AE\
Router\lib\commons-logging.jar;C:\Program Files\FileNet\AE\Router\lib\
commons-discovery.jar;C:\Program Files\FileNet\AE\Router\lib\jaxrpc.jar;C:\Program Files\
FileNet\AE\Router\lib\juddi.jar;C:\Program Files\FileNet\AE\Router\lib\addressing.jar;
C:\Program Files\FileNet\AE\Router\lib\Sandesha.jar;C:\Program Files\FileNet\AE\
Workplace\WEB-INF\lib\p8ciops.jar;C:\Program Files\FileNet\AE\Workplace\WEB-INF\
lib\javaapi.jar;C:\Program Files\FileNet\AE\Workplace\WEB-INF\lib\p8toolkit.jar;
C:\Program Files\FileNet\AE\Workplace\WEB-INF\lib\p8workplace.jar;
C:\Program Files\FileNet\AE\Router\lib;C:\Program Files\FileNet\AE\Workplace\WEB-INF"
  filenet.vw.integrator.base.VWComponentManager
/named
/routerURL hqruby_600
/userName=-1
/password ahuFjQT6oj5HekeMmh1zsB1Uddf=
/registryPort 32771
/eventPort 32773
/registryName FileNet.VW.VWComponentManager.hqruby_600.CE_Operations
/queues=CE_Operations
```

While there are the typical parameter, such as classpath and the PE API required properties (wasp.location and filenet.pe.bootstrap.ceuri), the following are worth a closer look:

- JAAS configuration file specified through `-Djava.security.auth.login.config`, for example:

```
"-Djava.security.auth.login.config=
C:/Program Files/FileNet/AE/Router/taskman.login.config"
```

By default, it is `taskman.login.config`. However, it can be any JAAS configuration file with the following requirement:

FileNetP8 stanza must be present containing the appropriate login module to connect to the CE depending on the CE transport protocol.

By default, when running outside of an application server, the Web Server Interface (WSI) transport protocol is recommended. Therefore, the `taskman.login.config` has the following:

```
FileNetP8
{
  com.filenet.api.util.WSILoginModule required debug=false;
};
```

In addition, the JAAS configuration file needs to contain the JAAS configuration context for the component queue. The JAAS context should have this line:

```
filenet.vw.server.VWLoginModule required;

For example:
Sample
{
  filenet.vw.server.VWLoginModule required;
  sample.module.SampleLoginModule required debug=true;
};
```

This explains the `CE_Operations` component queue that has `CELogin` as the JAAS configuration context. Therefore, the out-of-the-box (OOTB) `taskman.login.config` has:

```
CELogin
{
  filenet.vw.server.VWLoginModule required routerurl="hqruby_600";
  com.filenet.wcm.toolkit.server.operations.util.CELoginModule required credTag=Clear;
```

### Sample taskman.login.config

```
CELogin
{
  filenet.vw.server.VWLoginModule required routerurl="xx";
  com.filenet.wcm.toolkit.server.operations.util.CELoginModule required credTag=Clear;
};

HelloWorldLogin
{
  filenet.vw.server.VWLoginModule required;
```

```
};
FileNetP8
{
    com.filenet.api.util.WSILoginModule required debug=false;
};
```

Note that routerurl is no longer required for the VWLoginModule.

See the sidebar for a sample `taskman.login.config` file.

- Component Manager parameters are explained in Table 3.

**Table 3. Component Manager parameter descriptions**

<code>/named</code>	
<code>/routerURL=PEConnectionPointName</code>	Specifies the connection point name to access the PE
<code>/username=PEUsername</code>	PE Username
<code>/password=PEPassword</code>	PE Password
<code>/registryPort=32771</code>	Specifies the RMI registry port to register the Component Manager instance. The default value is 32771. This allows the AE Process Task Manager to manage the instance.
<code>/eventPort=32773</code>	The Component Manager can be set up to receive notification from the PE server when work arrives at any component queues. This is to avoid excessive polling. However, only one instance can be setup to receive notification. The default value is 32773.
<code>/registryName=&lt;Component Manager RMI registry Name&gt;</code>	Specifies the Component Manager RMI server name, must be unique within the same RMI registry. By default, when started by the AE Process Task Manager, it is: FileNet.VW.VWComponentManager.<connectionPointName> For example: FileNet.VW.VWComponentManager.hqruby_600.CE_Operat
<code>/queues=queueName</code>	Either * for ALL the queues or a specific queue or list of queues separated by comma.

/unbind	When present, this stops the current instance that has the RMI registry object with the specified /registryName.
---------	--

It is recommended that a command line to stop Component Manger should be used instead of the merciless terminate or kill. An instance of Component Manager can be stopped by re-issuing the same command line for starting with the additional **/unbind** option.

### Set up Eclipse to run Component Manager

As explained above, a JAAS configuration file is needed. Use the sidebar above as an example for the file, or copy the taskman.login.config located in the router directory and add the HelloWorldLogin JAAS context stanza:

```

HelloWorldLogin
{
  filenet.vw.server.VWLoginModule required;
};
    
```

Put this file to a directory, for example: C:\SampleWS\config.

Duplicate the PEComponentQueueHelper configuration in the Run Dialog, and set:

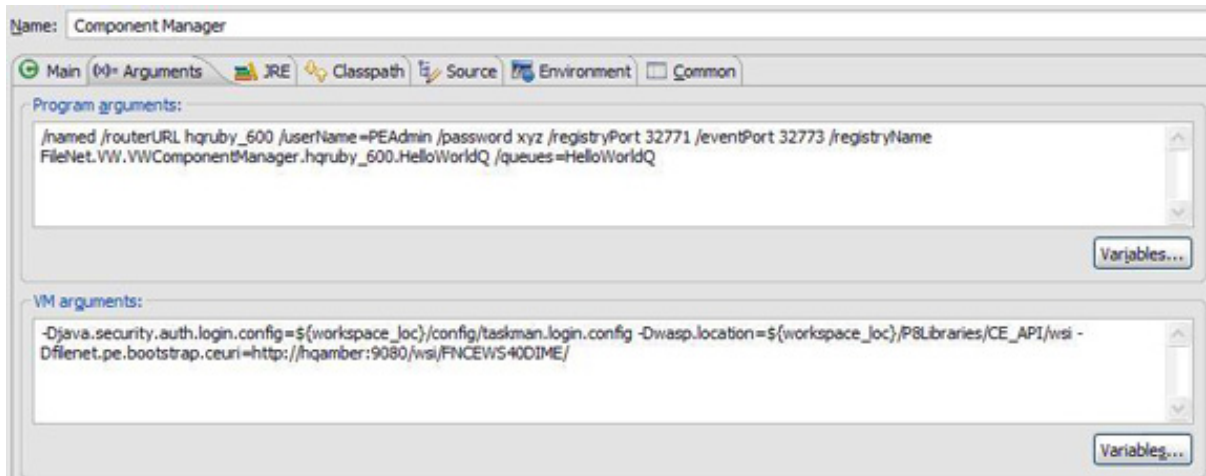
- **Component Manager** for Name
- **filenet.vw.integrator.base.VWComponentManager** for Main Class

On the Arguments tab, set the parameters according to Table 4. See [Figure 4](#) for examples.

**Table 4. Component Manager parameters in Eclipse**

Program arguments	/named /routerURL <i>PEConnectionPoint</i> /userName= <i>PEUser</i> /password <i>PEPassword</i> /registryPort 32771 /eventPort 32773 /registryName FileNet.VW.VWComponentManager. HelloWorldQ /queues=HelloWorldQ
VM Arguments	-Djava.security.auth.login.config=\${workspace_loc}/con -Dwasp.location=\${workspace_loc}/P8Libraries/CE_API/ws -Dfilenet.pe.bootstrap.ceuri=http://CEServer:CEPort/ws

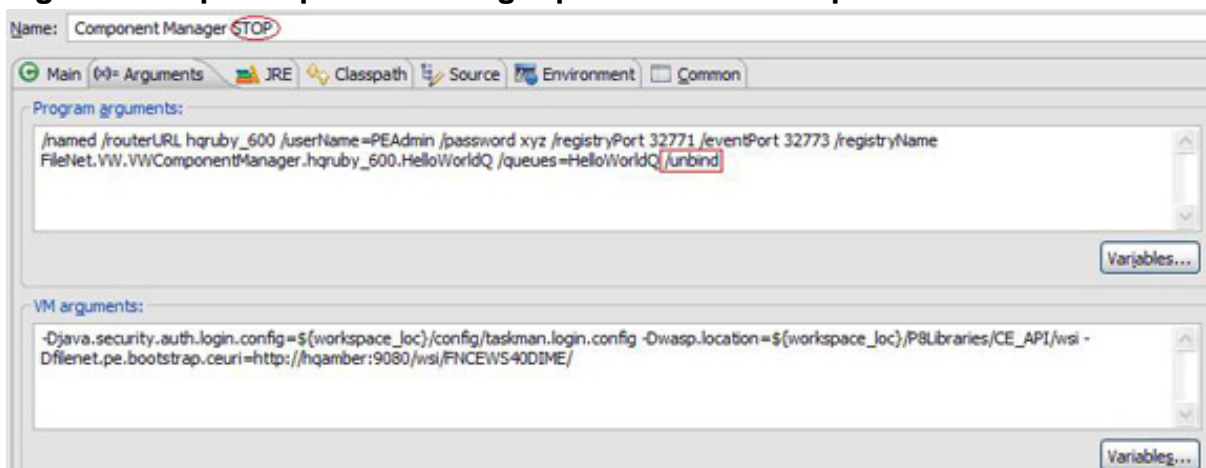
**Figure 4. Example of Component Manager parameters in Eclipse**



Create a run configuration to stop Component Manager by duplicating the “Component Manager” run configuration, and name it **Component Manager STOP**.

Add `/unbind` to the program arguments. See Figure 5.

### Figure 5. Stop Component Manager parameters in Eclipse



Before debugging the Component Manager, make sure that you add a breakpoint in the `sayHello` method and **make sure that all other Component Manager instances for the same PE connection point have been stopped**. Otherwise, work in the queue is picked up and processed by those instances.

As the Component Manager processes the work items in the `HelloWorldQ` and invokes the `sayHello` method for the step, you can then examine the values of the parameters or single step through the instructions in the method easily.

Run the **Component Manager STOP** configuration to stop the running Component Manager instance.

## Conclusion

This article shows tips and shortcuts in the development process of a custom component for the IBM FileNet BPM. With this knowledge, tracing is not the only choice for debugging the component while it is in development, thus shortening the time to production. In addition, the sample steps show in details how other P8 applications can be set up for debugging in Eclipse with the P8 PE Component Manager as an example. Users of other Java IDEs could use this information to set up their development environment for debugging P8 applications as well.

# Resources

## Learn

- [FileNet Business Process Manager](#) page: Find more information about the IBM FileNet Business Process Manager.
- [Java Authentication and Authorization Service \(JAAS\) Reference Guide](#): Discover information regarding the JAAS framework.
- [developerWorks Information Management zone](#): Learn more about Information Management. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with [developerWorks technical events and webcasts](#).
- [developerWorks Architecture zone](#): Get the resources you need to advance your skills in the architecture arena.
- Browse the [technology bookstore](#) for books on these and other technical topics.

## Get products and technologies

- Eclipse can be downloaded at the [Eclipse](#) home page.
- Download [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from IBM Information Management, Lotus®, Rational®, Tivoli®, and WebSphere®.

## Discuss

- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

## About the authors

Dao-Quynh L. Dang

Dao-Quynh is a senior software developer for P8 BPM. She is the architect for the P8 BPM Integration tier including Web services, process orchestration, the Web services API, and the Component Integrator.

---

Indrajit Poddar

Indrajit Poddar (IP) is a member of the Strategy, Technology, Architecture and Incubation (STAI) team in the IBM Software Group, where he is architecting a

number of PoCs for building SOA composite business services using IBM SOA Foundation products.