

DB2 Content Manager / Process Choreographer Integration Quick Start

Revision1

DB2 Content Manager Enterprise Edition Version 8.3 introduced a tutorial and sample code "Integrating with business processes" using the Process Choreographer component of *WebSphere Business Integration Server Foundation V5.1.1*. This Integration Quick Start Guide replaces the tutorial and describes Revision 1 of this integration based on the following chapters:

1. [Introduction](#)
2. [Setting up the Development Environment and Testing the Sample Process](#)
3. [A Tour of the Sample Process](#)
4. [Developing a Custom Content Manager / Process Choreographer Integration Solution](#)
5. [Deploying an Integrated Solution on a Production Simulation Environment](#)
6. [Hints and Troubleshooting](#)
7. [Further Reading](#)

1. Introduction

DB2 Content Manager Enterprise Edition V8.3 includes a high-performance document routing engine to manage the lifecycle of content objects. For complex process environments that involve a variety of applications and services in a heterogeneous IT and business environment *WebSphere Business Integration Server Foundation Version 5.1.1* is the process management environment of choice as it provides a rich infrastructure for intra-enterprise and inter-enterprise services based on the WebSphere Application Server. These processes can involve both human and IT resources. The types of processes can vary greatly, ranging from Web services or Web page navigation to business transaction support. Processes can be based on automated steps that together constitute a single transaction or they can involve human interaction and may run for days, weeks or months.

The Content Manager / Process Choreographer (CM/PC) Integration Quick Start provides some key ingredients for the creation of content-centric processes. The term content-centric process refers to a process in which a content object stored in the Enterprise Content Management system plays an important role. This typically means that the content object may be investigated at human activities and that it is modified throughout the process. This content object (a Content Manager item) which typically is a container of content objects (a folder) is said to be *routed through the process or (work)flow*. Modifying it can mean a number of things such as adding or removing objects from it (if it is a folder), changing attribute values, modifying access rights, adding annotations, signatures, or watermarks, etc.

The main purpose of the Integration Quick Start is to provide best practices and pre-assembled building blocks that support the creation and management of content-centric processes based on the Process Choreographer environment.

1.1 What's New in Revision 1

Revision 1 of the Content Manager / Process Choreographer Integration Quick Start consists of the following elements:

- *Integration Toolkit*
- *Quick Start Client*
- *Auto Claims Process*

The Integration Toolkit is a collection of service components (Java classes, EJBs) that implement the following interaction patterns between the content management and process management system:

- *Content Event Handling* ... item creation or deletion events in Content Manager trigger corresponding actions in Process Choreographer
- *Collection Point* ... let a process instance wait until the folder that is routed through the process contains a certain number of items of a certain type or until a timeout or exception occurs.
- *Content Attribute Access* ... efficiently access the value of content attributes from within the business process, e.g. to enable content-specific decisions (Decision points)
- *Folder Service* ... add the item routed through the process to a retrieved folder or add a retrieved item to the folder routed through the process
- *Combined Query* ... efficiently retrieve process information from Process Choreographer and associated content attribute values from Content Manager
- *Common Staff Repository* ... map Content Manager user and group definitions to the Process Choreographer staff resolution facility based on a WebSphere® custom user registry

The Quick Start Client is a Web application that enables process users to see work that is assigned to them, fetch work items and work with content objects in DB2® Content Manager Enterprise Edition V8.3. Though the Quick Start Client is customized for use with the Auto Claims Process it provides generic out of the box capabilities and can easily be adjusted to the needs of other processes.

The Auto Claims Process illustrates how the Integration Toolkit can be used to create a content-aware process. This process involves human-based activities that can be performed using the Quick Start Client, services provided by the Integration Toolkit and some simulated internal or external Web services that may involve legacy applications or B2B transactions.

Both, the Integration Toolkit and the Quick Start Client are provided as sample code. They also serve as educational resources for learning how to integrate the two products. The Quick Start Guide that can be downloaded below describes how the source code can be modified to meet specific requirements of a custom integration solution. See the license statements in the documentation or source files for details.

1.2 Notices, License and Support Terms

Note that the following statements apply to the source code of the Content Manager / Process Choreographer Integration Quick Start:

```
Licensed Materials - Property of IBM
IBM DB2 Content Manager Enterprise Edition V8 (program number 5724-B19)
(c ) Copyright IBM Corp. 1994, 2005. All Rights Reserved.

US Government Users Restricted Rights
Use, duplication or disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.

DISCLAIMER OF WARRANTIES :

Permission is granted to copy and modify this Sample code, and to distribute
modified versions
provided that both the copyright notice, and this permission notice and warranty
disclaimer
appear in all copies and modified versions.

This software is provided "AS IS." IBM and its Suppliers and Licensors expressly
disclaim all
warranties, whether EXPRESS OR IMPLIED, INCLUDING ANY IMPLIED WARRANTY OF
MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE OR WARRANTY OF NON-INFRINGEMENT.
IBM AND ITS SUPPLIERS AND LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFEREDBY
LICENSEE
THAT RESULT FROM USE OR DISTRIBUTION OF THE SOFTWARE OR THE COMBINATION OF THE
SOFTWARE WITH
ANY OTHER CODE.
IN NO EVENT WILL IBM OR ITS SUPPLIERS AND LICENSORS BE LIABLE FOR ANY LOST REVENUE,
PROFIT OR
DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE
DAMAGES, HOWEVER
CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR
INABILITY TO USE
SOFTWARE, EVEN IF IBM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
```

Note that the integration toolkit requires a DB2 Universal Database-based library server and that it does not support WebSphere Process Server V6, the z/OS or i5/OS environments, nor clustered environments.

See [Trademarks](#) for details on the trademarks used in this document.

1.3 Software Requirements

For both the development and runtime environments, the Content Manager library server needs to be based on a *DB2 Universal Database V8.2*. The integration toolkit only supports a DB2 Universal Database-based library server.

The following system configuration is required for the development and test environment ([chapter 3](#)):

- a. Development workstation A hosts:
 - *WebSphere Studio Application Developer Integration Edition V5.1.1*

- *DB2 Universal Database V8.2 Runtime Client*
 - *DB2 Content Manager V8.3 Client for Windows*
 - *DB2 Information Integrator for Content V8.3 - Content Manager version 8 connector and Java connector toolkit*
- b. Repository server B with:
- *DB2 Content Manager Enterprise Edition V8.3*

For simulating a production environment ([chapter 5](#)) the repository server B additionally requires:

- *WebSphere Business Integration Server Foundation V5.1.1*
- *Tivoli Directory Server V5.2.2 or V5.3*

The development workstation as well as the server should have at least 1 GByte of main memory. However, recommended are at least 2 GByte.

This guide uses Windows platform conventions for path names. Before proceeding we recommend that you look up the install directories of the following products as they will be used later in this guide:

| | | |
|---------------|---|--|
| <IBMCMROOT> | = | Content Manager installation folder (e.g. c:\Program Files\IBM\db2cmv8 or /usr/db2cmv8) |
| <DB2HOME> | = | home folder of the DB2 instance which hosts the Content Manager library server (e.g. c:\Program Files\IBM\sqlllib or /home/db2inst1/sqlllib) |
| <WSADIE_HOME> | = | WebSphere Studio Application Developer Integration Edition install folder (e.g. c:\Program Files\IBM\WebSphere Studio\Application Developer IE\v5.1.1) |
| <WAS_HOME> | = | WebSphere Application Server install folder (e.g. c:\Program Files\IBM\WebSphere\Appserver) |

1.4 A guideline for reading this document

We recommend starting with [2. Setting up the Development Environment and Testing the Sample Process](#) as this lays the foundations for further explorations in the area of integrating content and workflow. This chapter describes how to set up the infrastructure to build and test integrated workflows. You may want to take a tour of the sample process as described in [3. A Tour of the Sample Process](#) and continue either with [4. Developing a Custom Content Manager / Process Choreographer Integration Solution](#) to understand what it means to create a custom CM /PC integration solution based on the Integration Toolkit and Quick Start Client or to [5. Deploying an Integrated Solution on a Production Simulation Environment](#) which explains how the sample process can be deployed on a production environment. [6. Hints and Troubleshooting](#) provides some hints and tips which may be useful when working with the sample. [7. Further Reading](#) provides links to product documentation, Redbooks and Support pages.

2. Setting up the Development Environment and Testing the Sample Process

This chapter describes the steps needed to set up a development environment for the Integration Quick Start. At the end of this chapter the Quick Start Client can be used to work the sample process and to explore the results of running the server-side integration functions. Depending on the reader's previous experience with the WebSphere Studio Application Developer environment completing this paragraph will take approximately 1-2 hours.

2.1 Prerequisites and Installation Hints

Detailed instructions on how to install and configure the required software can be found in the product documentation. See [7. Further Reading](#) at the end of this document. This section gives some hints about configuration settings that are specific to the use of these products in the context of the Integration Quick Start. See [1.3 Software Requirements](#) for details on the system configuration required for chapter 1.

Installation hints for Information Integrator for Content

In each of the following installation windows, perform the described task:

- **Installation Destination**
Verify that the target directory is set to <IBMCMROOT>.
- **Setup Type**
Select *Custom*.
- **Select the components that you want to install**
Select *DB2 Content Manager Version 8 connector and DB2 Content Manager Version 8 Java connector toolkit*.

- **Working Directory**

Keep the default <IBMCROOT>.

- **Server Connection Configuration**

Select **Configure RMI for remote connection to the above selected servers** to set up the connection from the Development Workstation to the Content Manager server.

Installation hints for WebSphere Application Developer Integration Edition

- No special hints are required: follow the steps described in the product documentation.

Note that *WebSphere Business Integration Server Foundation* and *Tivoli Directory Server* are not required in the development and test environment.

2.2 Preparing the Content Manager Sample Data for use with the Sample Process

The Quick Start sample process is based on a modified version of the Content Manager sample data. This step describes how to load the sample data and adjust it for use with the sample process. The sample process uses a new attribute `XYZ_ClaimAmount` to distinguish 'cheap' cases for which a high process throughput is desirable from 'expensive' cases for which additional research is required. Furthermore, two new attributes `WF_OnCreate` and `WF_OnDelete` are required to enable creation and termination of process instances based on content events (see [Content Event Handling](#) for details).

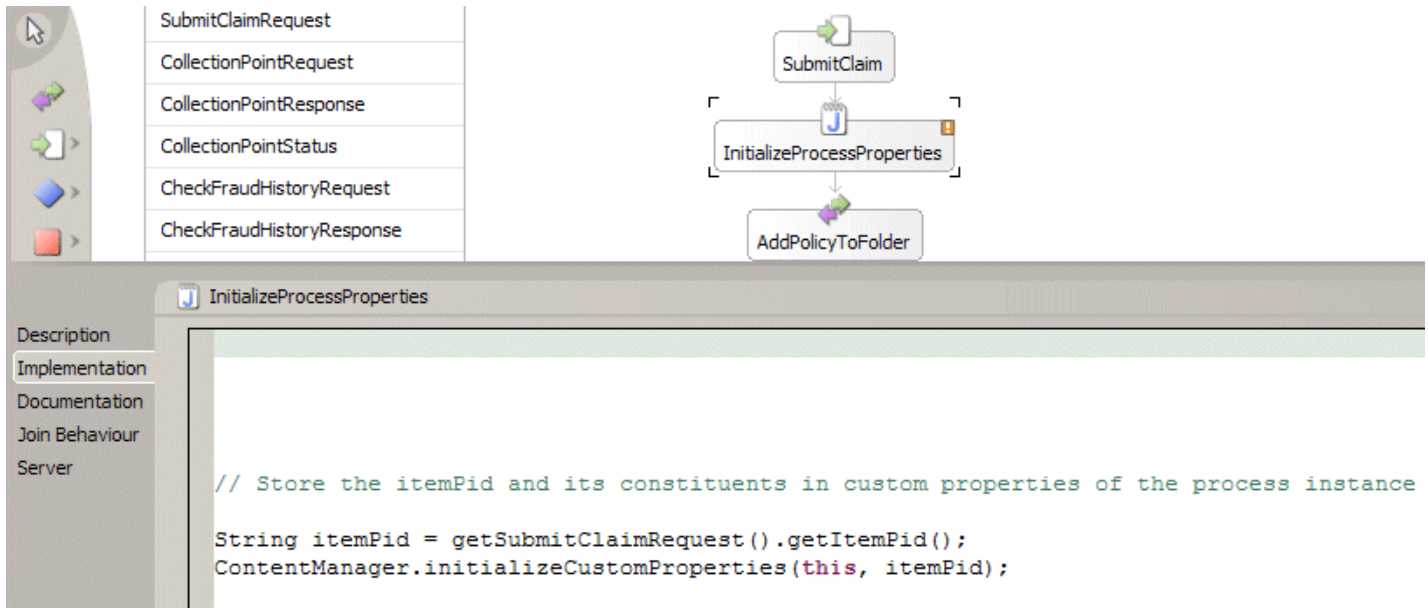
- Start the Content Manager First Steps program.
- If you have worked with the sample data before, click **Remove Sample Data** to ensure all data is in a clean state.
- Click **Load Sample Data**.
- Wait until the sample data creation completes.
- Click **Work with Sample Data** to start the Content Manager Administration client.
- Log on to the administration client with your library server administrator userid and password.
- Select **Data Modelling**, right-click **Attributes**, select **New** to define a new attribute `XYZ_ClaimAmount` with the properties shown below:

The screenshot shows a 'New Attribute' dialog box with the following fields and options:

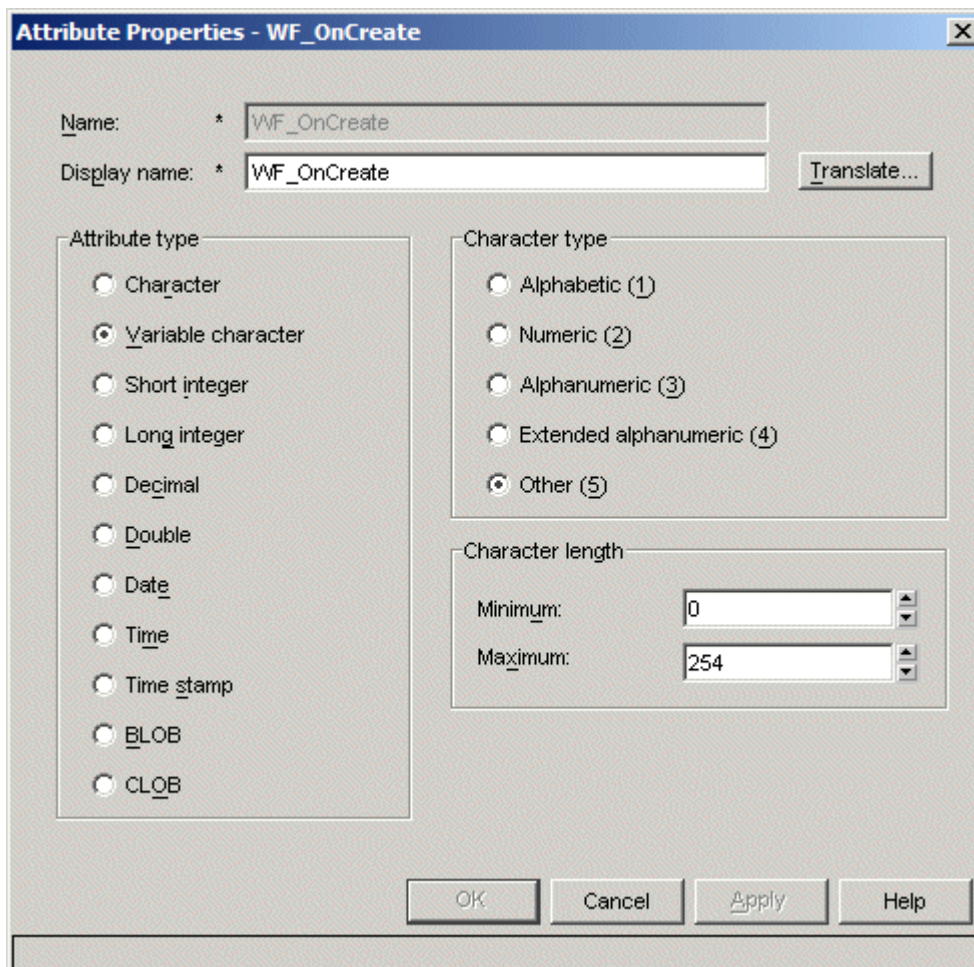
- Name:** * XYZ_ClaimAmount
- Display name:** * Claim Amount (with a **Translate...** button)
- Attribute type:** A list of radio buttons with 'Decimal' selected:
 - Character
 - Variable character
 - Short integer
 - Long integer
 - Decimal
 - Double
 - Date
 - Time
 - Time stamp
 - BLOB
 - CLOB
- Decimal length:** A sub-dialog with two spinners:
 - Total:** 12
 - Fixed places:** 2
- Buttons:** OK, Cancel, Apply, Help

- Click **OK** to store the new attribute.

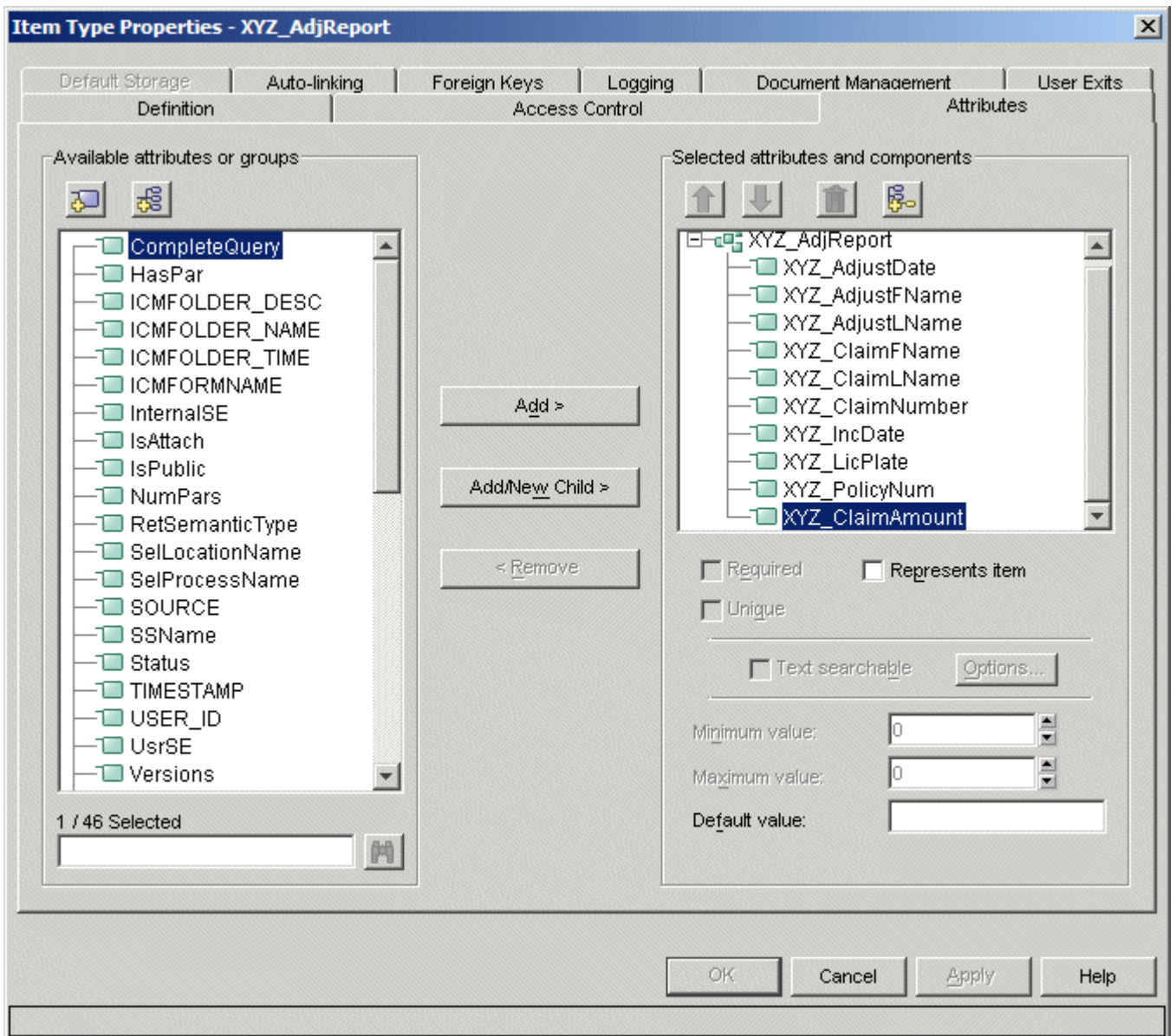
- Right-click **Attributes**, select **New** to define a new attribute WF_OnCreate with the properties shown below and click **OK** to store the new attribute:



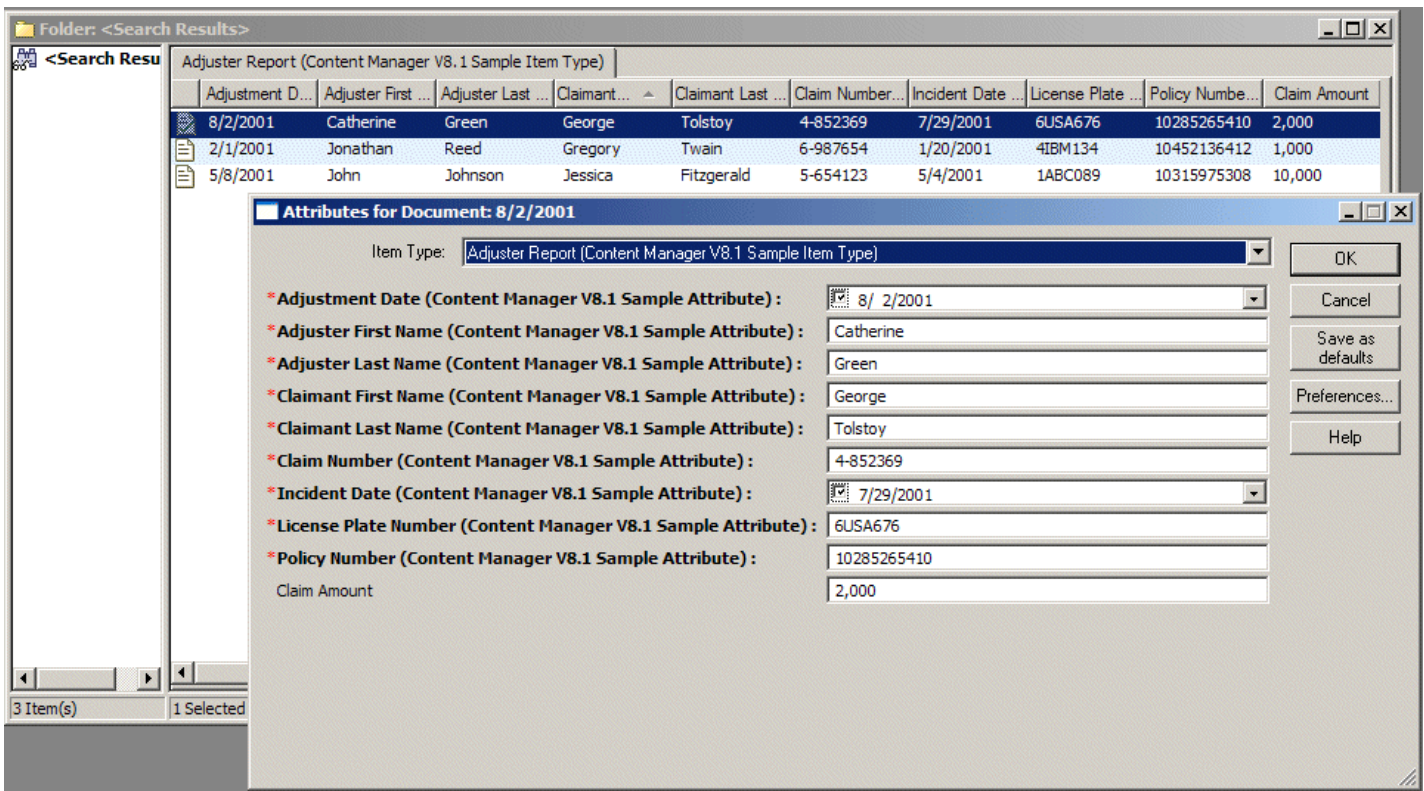
- Repeat the previous step to define a new attribute **WF_OnDelete** with the same characteristics as **WF_OnCreate**.
- In the Data Modelling view, expand **Item Types**, right-click on XYZ_ClaimFolder, right-click **Properties**, open the **Attributes** tab and add two new attributes: WF_OnCreate and WF_OnDelete both with the default value ClaimsHandlingProcess



- Right-click XYZ_AdjReport and select **Properties**.
- On the definition tab change the **New version policy for attributes** to **Never create**.
- Switch to the **Attributes** tab and add the attribute XYZ_ClaimAmount to this item type:



- Close the System Administration Client.
- Close the First Steps program.
- Start the *DB2 Content Manager Client for Windows* and log on as library server administrator (icmadmin). Note that if the *Content Manager Client for Windows* was already started while you changed the sample data definitions in the previous step you need to close and restart it so the new definitions are available.
- Select **Search -> Basic** and locate all instances of item type Adjuster Report (Content Manager v8.1 Sample Item Type).
- Right-click on each of the three entries in the result list, select **Attributes** and enter different Claim Amount values to the three items: for example 1000 (small claim), 10000 (large claim) , 2000 (small claim) as shown below. The sample process introduced later will handle documents with a small claim amount (i.e. less than or equal to \$5000) differently:



- Close the *DB2 Content Manager Client for Windows*.

2.3 Setting up the Test Environment

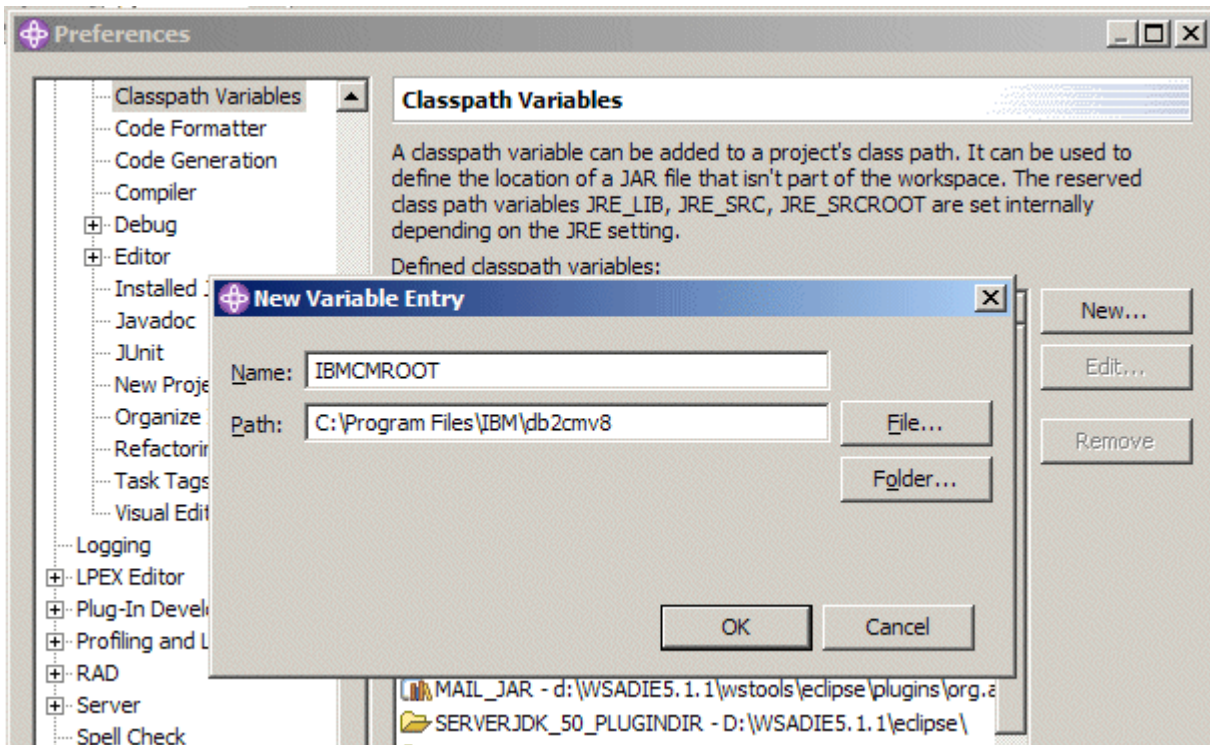
This section copies the files that are required for the build environment to the appropriate locations of the development workstation's file system. Two JAR files need to be located in the `lib` directory of the application server in order to be accessible from Java snippets in business processes. The build environment uses a file-based custom user registry for authentication with the process environment (via the application server). This user registry stores userIDs and passwords of the process staff in two files that need to be placed into a folder in the application server install directory. The settings in the password file (especially for userID `icmadmin`) need to be adapted to the settings of the Content Manager server. Note that the runtime environment uses a common LDAP-based user repository for both Content Manager and the process environment so there will be no duplication of credentials and passwords will not be stored in readable form when moving to the runtime environment. The file-based approach is only used for the development environment.

- Unzip `quickStartSampleRev1wbi.zip` into a temporary folder `<TMP>`.
- Copy the files `bpecm.jar` and `bpecmutil.jar` from `<TMP>\runtime` to `<WSADIE_HOME>\runtime\ee_v51\lib`.
- Open `<TMP>\runtime\bpecm.properties` in an editor and make sure the property `ContentManagerAuthenticationPwd=...` contains the password of the library server administrator (`icmadmin`). Note that a simple encryption method referred to as ROT13 can be used to avoid storing the password unencrypted. Apply ROT13 encryption by adding a decimal 13 to the numeric character representation of each character in the password string, e.g. ABC would become NOP. Set `ContentManagerAuthenticationPwdEncryption` to 1 to enable the use of ROT13 encryption. Note that the proposed encryption is for demo purposes only. If you plan to use the code in a production environment you might want to choose a more enhanced encryption method.
- Copy the file `bpecm.properties` from `<TMP>\runtime` to `<WSADIE_HOME>\runtimes\ee_v51\properties`.
- Open `<TMP>\runtime\users.prop` in an editor, locate the line `icmadmin:passwd:2:2:` and replace `passwd` with the password of the library server administrator (`icmadmin`) on your system.
- Create a folder `security` under `<WSADIE_HOME>\runtimes\ee_v51` and copy the files `users.prop` and `groups.prop` from `<TMP>\runtime` to this new directory.
- Create a folder in your file system that will serve as the workspace folder for WebSphere Application Developer Integration Edition referred to as `<WORKSPACE_FOLDER>` later in this document.
- Recursively copy all contents of `<TMP>\projects\` into this directory so that it contains the folders `ClaimsHandlingProject`, `ContentManagerIntegration`, etc. and all files located in these folders.

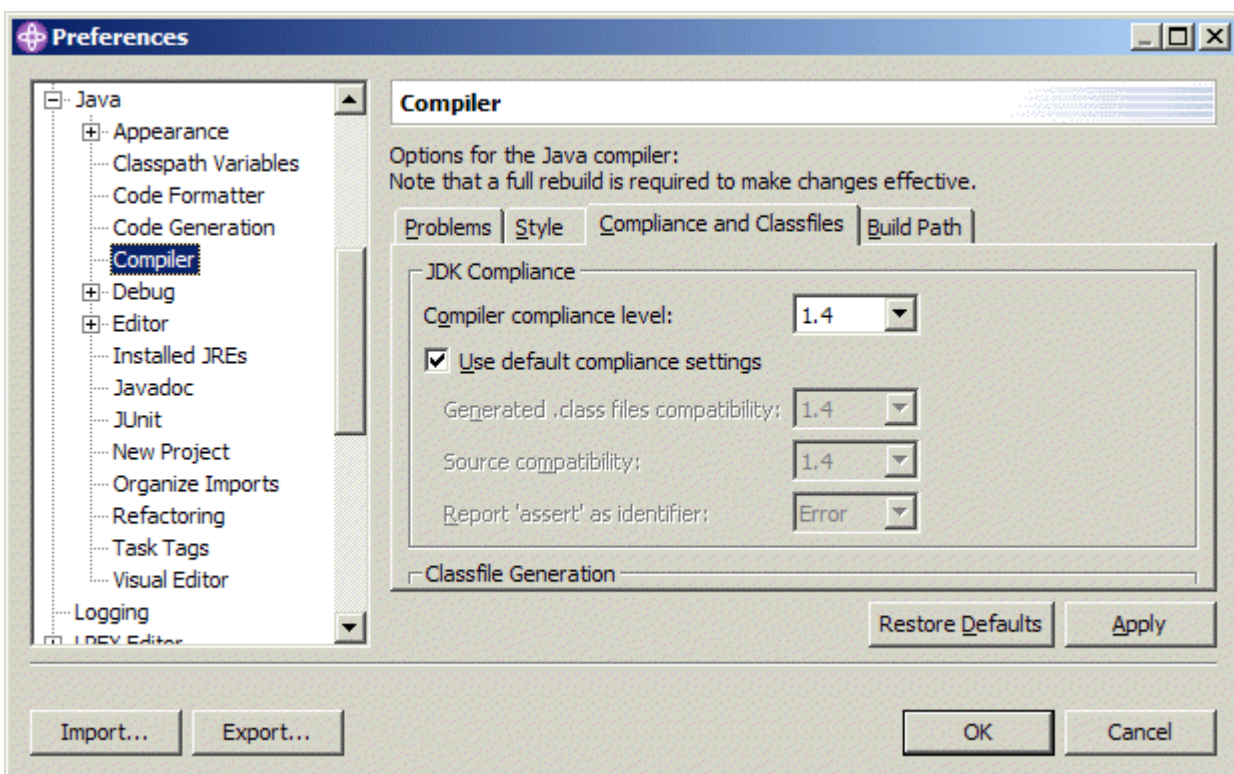
2.4 Configuring the Build environment and Importing the Sample Projects

This step adjusts compiler settings and makes the project files available to the build environment. We recommend setting the browser configuration so it opens the Web browser in a new window which provides more flexibility when watching project contents and running the Web client at the same time.

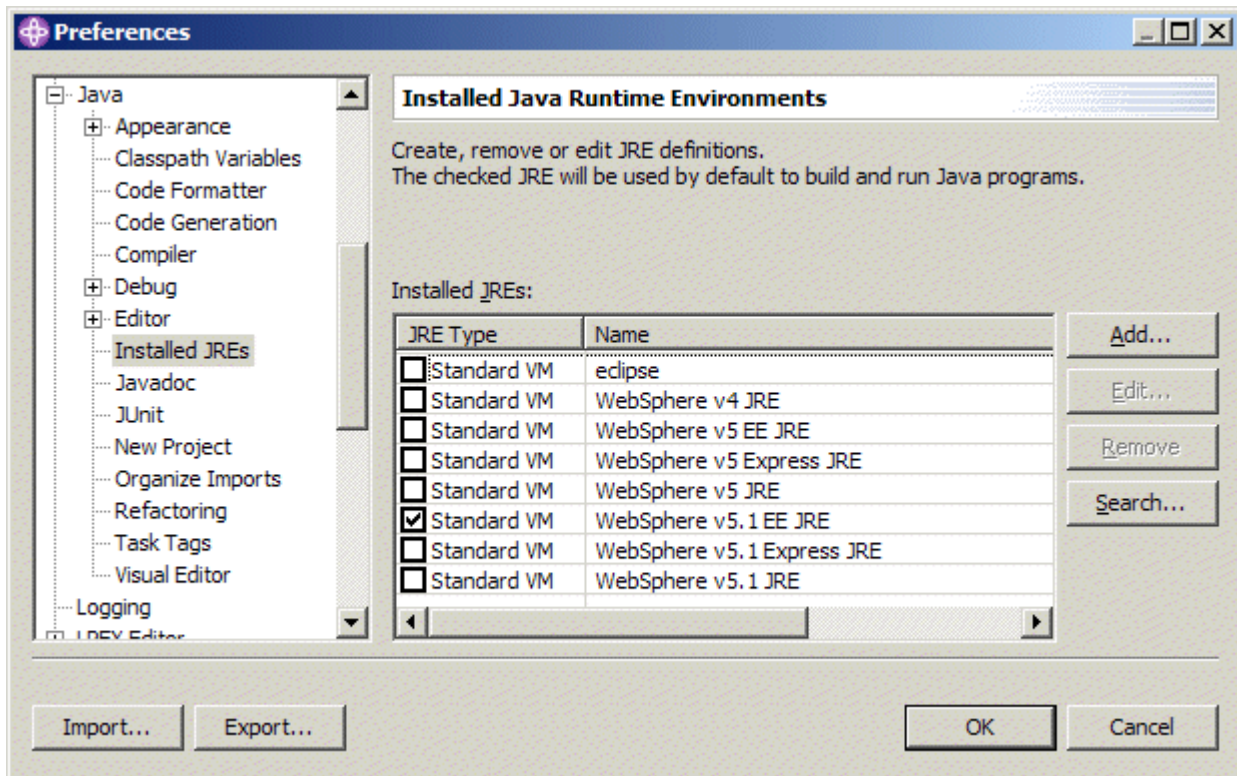
- Start *WebSphere Studio Application Developer Integration Edition* and specify the folder created in the previous step as your workspace. If not prompted for a workspace, start it from the command line as follows: `%WSADIE_HOME%\wsappdevie.exe -setworkspace.`
- Select **Window > Preferences**. The Preferences window opens.
- Select **Java > Classpath Variables** and click the button labeled **New**. The 'New Variable Entry' window opens.
- Enter `IBMCMROOT` into the 'Name' field and the value of `<IBMCMROOT>` into the 'Path' field.




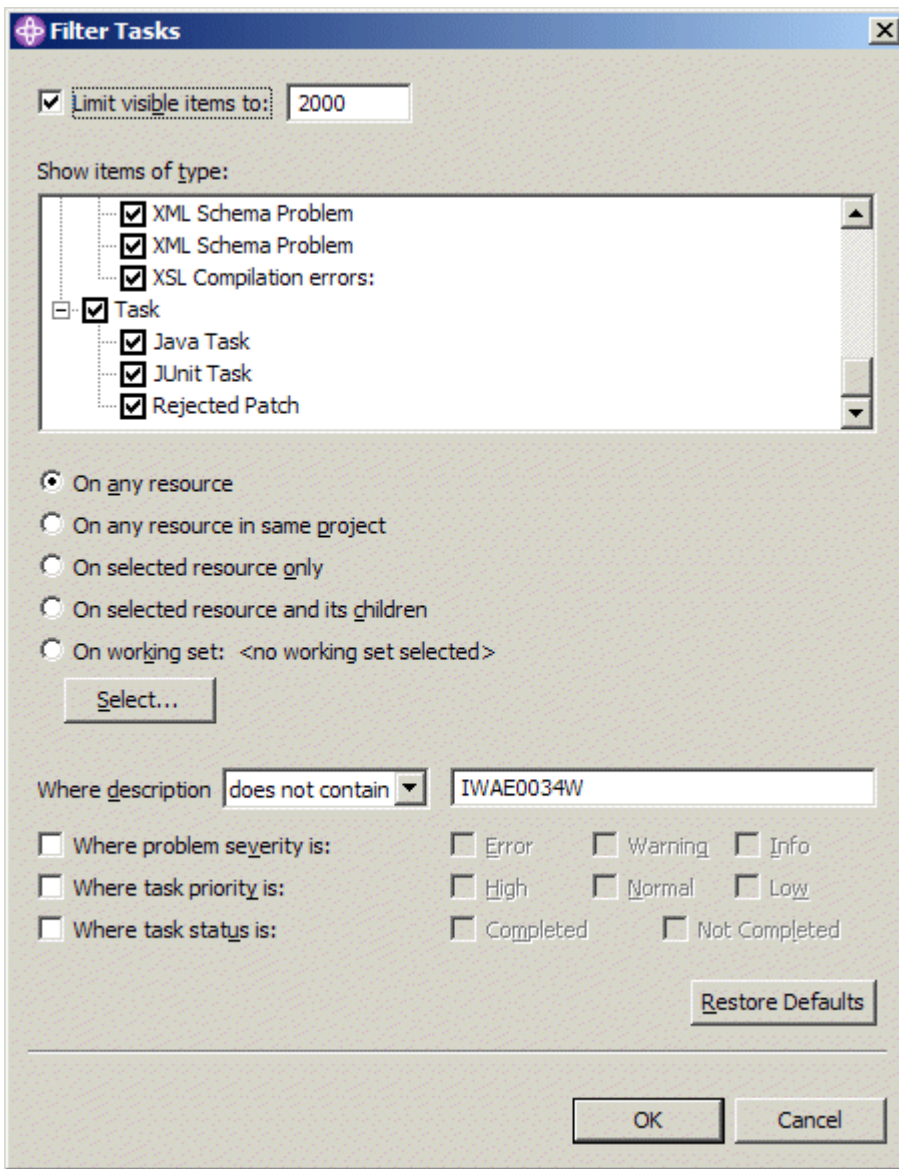
- Select **Java > Compiler**, click the **Compliance and Classfiles** tab and change the **Compiler compliance level** from default 1.3 to 1.4



- Select **Java > Installed JREs** and change the JVM from default **Eclipse** to **WebSphere v5.1 EE JRE**



- Recommended: select **Web Browser** and click **Use external Web Browser** and make sure the **Location** field points to the browser executable.
- Click **OK** in the preferences window to make the changes permanent. On the pop-up window 'The compiler settings have changed. A full rebuild is required...' click **No** as there are no projects yet that need to be rebuilt.
- Perform the following steps on the folders `ClaimsHandlingProject`, `ContentManagerIntegration`, `ContentManagerIntegrationEjb`, `ContentManagerIntegrationJar`, `QuickStartClient`, `Servers` located in your workspace. Any errors or warnings that might show up in the task view may temporarily be ignored. They should disappear when the deploy code has been generated by the end of this task.
 - Select **File > Import** from the main menu. The Import window opens.
 - Select **Existing Project into Workspace**. Click **Next**. The Import window opens.
 - Click **Browse...**, select the appropriate project folder and click **OK**.
 - Click **Finish**.
- Switch to the **Services** view of the business integration perspective.
- Expand **Deployable Services**, right-click `ContentManagerIntegrationEjb`, and select **Generate > Deployment and RMIC Code**. This opens the 'Generate Deployment and RMIC Code' window, On this window choose **Select All** and click **Finish**.
- On the menu select **Project > Rebuild All**.
- The three information messages `CHKJ2500I ... must be serializable at runtime can safely be ignored`.
- In the **Services** view expand **Service Projects > ClaimsHandlingProject > com.ibm.bpe.cm.sample**, right-click `ClaimsHandlingProcess.bpel` and select **Enterprise Services > Generate Deploy Code**. The 'Generate BPEL Deploy Code' window opens.
- Click **OK** to start deploy code generation. Wait until the progress information window disappears.
- The 9 remaining messages in the task view (`IWAE0034W EJB link element ... is unresolvable in module...`) can safely be ignored. Alternatively, they may be filtered by clicking the **Filters** button () in the task view menu, selecting `does not contain` in the box following **Where description** and entering `IWAE0034W` in the input field.



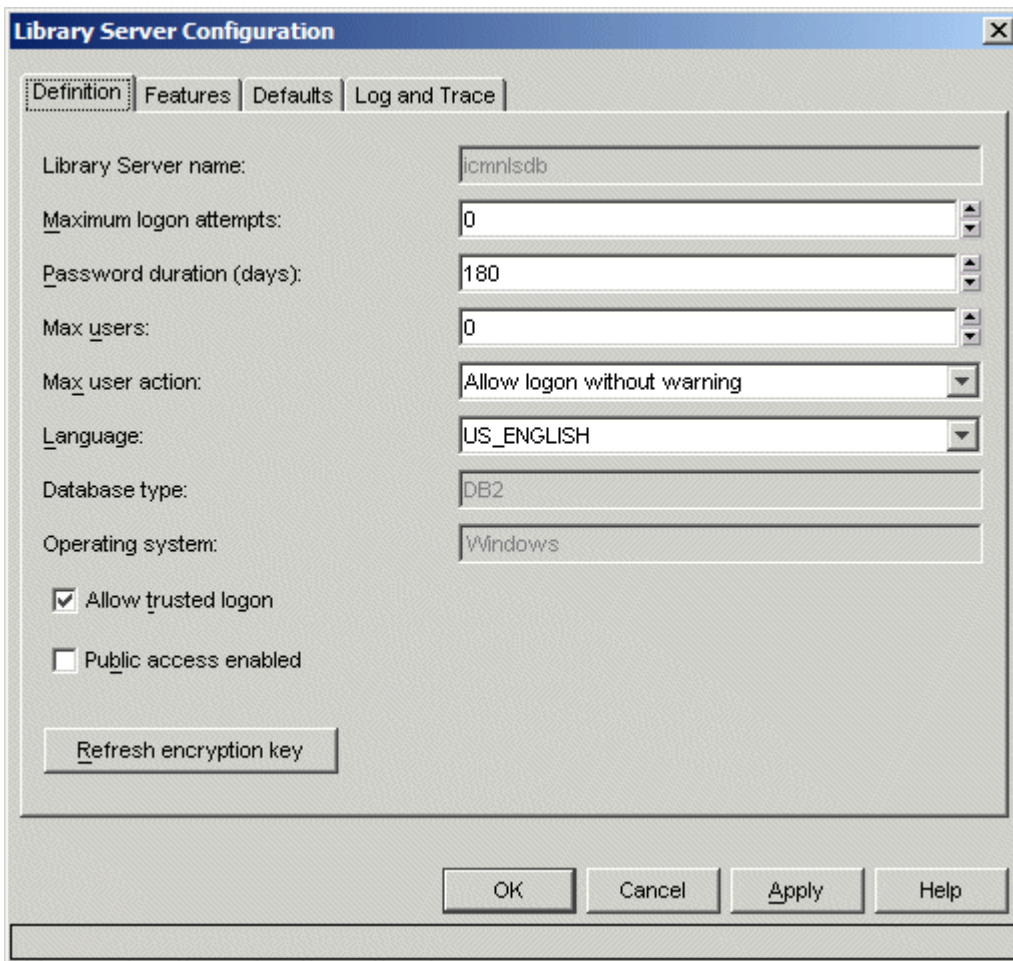
If **Fix Pack 1** of DB2 Content Manager Enterprise Edition V8.3 is installed we recommend taking the step [Upgrading the content viewer applet to Fix Pack 1 or \(re-\)creating the content viewer applet](#) to upgrade to the latest version of the content viewer applet.

You may now want to spend a moment and investigate the process and its elements. Useful resources are the process definitions (ClaimsHandlingProcess.bpel, ClaimsHandlingProcess.wsdl, ClaimsHandlingProcess.component) located in the Service Project ClaimsHandlingProject.

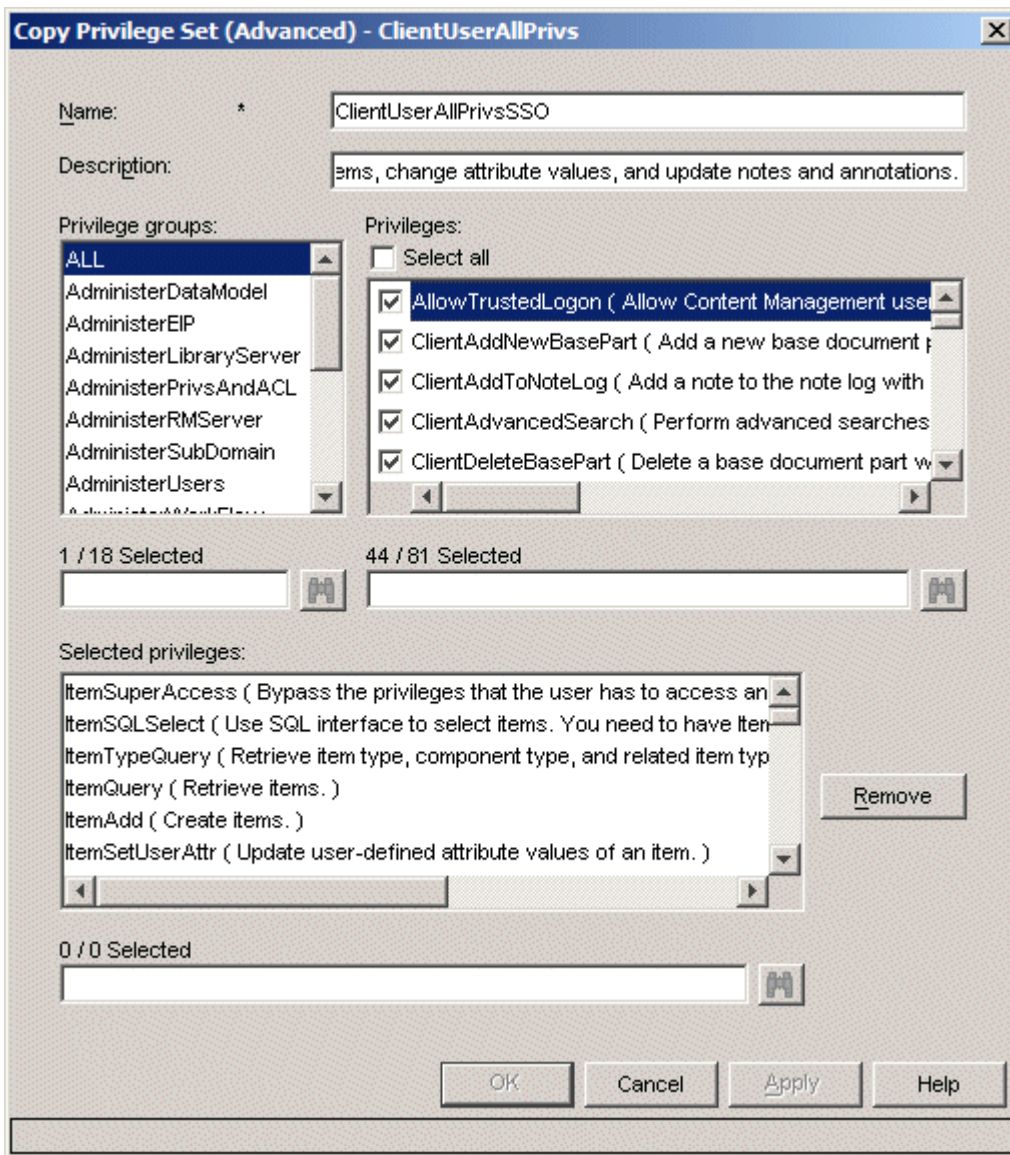
2.5 Enabling Content Manager for Single Sign On

The Quick Start Client uses Single Sign On (SSO) so users need to authenticate only once instead of having to logon to the process environment and to the Content Manager server. This step configures Content Manager so it can use the credentials provided by the application server to authenticate a user. Note that some of these settings are global (allow trusted logon, password not required for all users) so it is important to understand their potential impact on other Content Manager-based applications.

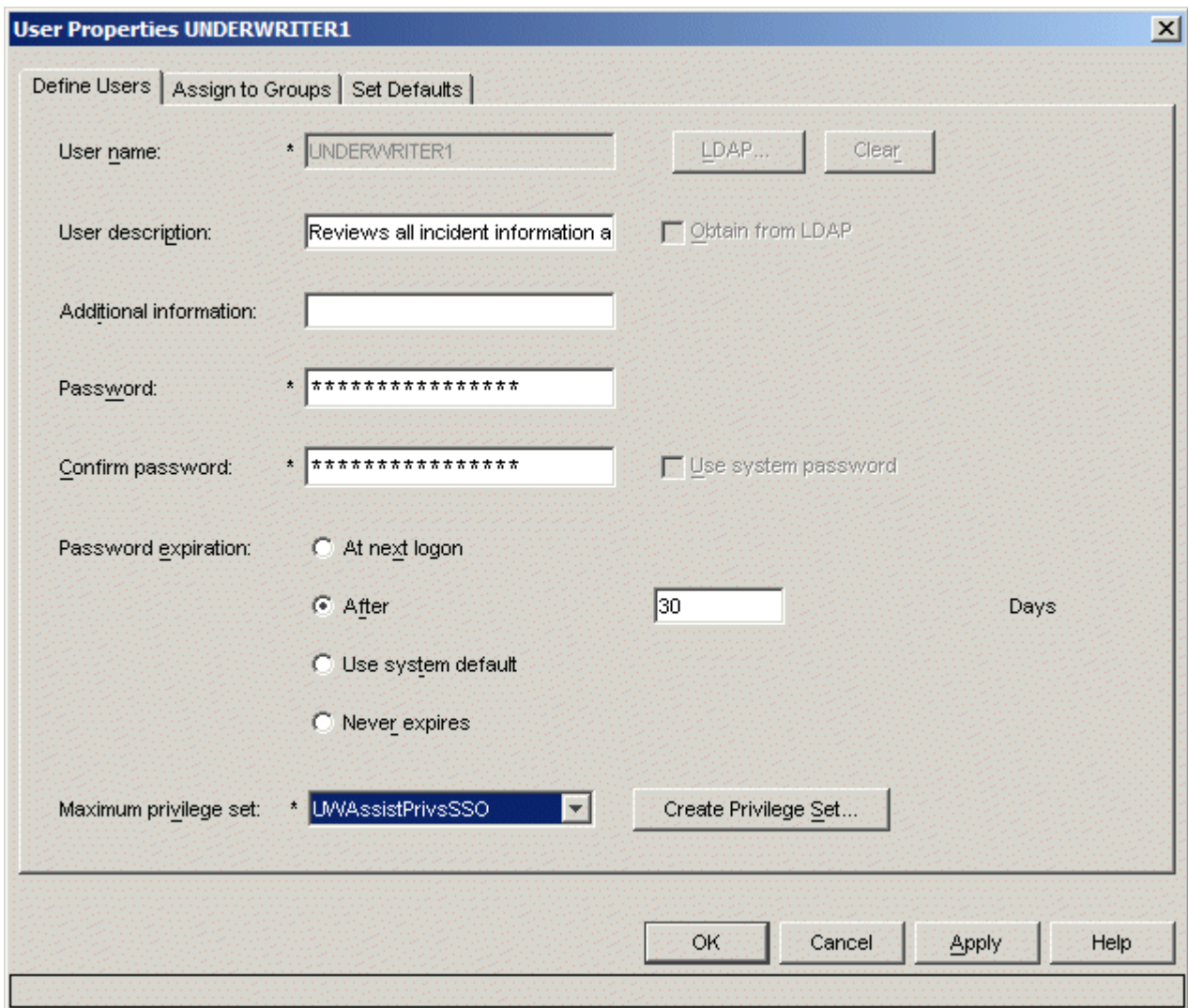
- Start the *DB2 Content Manager System Administration Client* and log on as library server administrator (i.e. icmadmin).
- In the navigation panel, click **Library server parameters > Configurations**.
- Right-click Library Server Configuration in the configuration window on the right and select **Properties**.
- Set **Max user action** to **Allow logon without warning** and select the **Allow trusted logon** check box



- Click **OK** to save the changes.
- Select **Tools > Manage Database Connection ID > Change Database Shared Connection ID** from the menu. The **Change Shared Database Connection ID and Password** window opens.
- Enter the connection user password (The default connection user is icmconct). Clear the **Password is required for all users** check box.
- Click **OK** to save the change.
- Click **Authorization > Privilege Sets**. A list of privilege sets shows up in the right pane.
- Right-click **ClientUserAllPrivs** and select **Copy > Advanced**. The Copy Privilege Set window opens.
- Enter the name `ClientUserAllPrivsSSO`, select **AllowTrustedLogon** in the Privileges field and click **OK** to save the new privilege set.



- In the navigation panel click **Authentication > Users**. The right pane now shows the list of DB2 Content Manager users currently defined.
- For each of the users Agent1, Agent2, Adjuster1, Adjuster2, Underwriter1, Underwriter2, UnderwriterAssistant1, and UnderwriterAssistant1 double-click the name to open the properties window and replace the value in the Maximum Privilege set field with ClientUserAllPrivsSSO

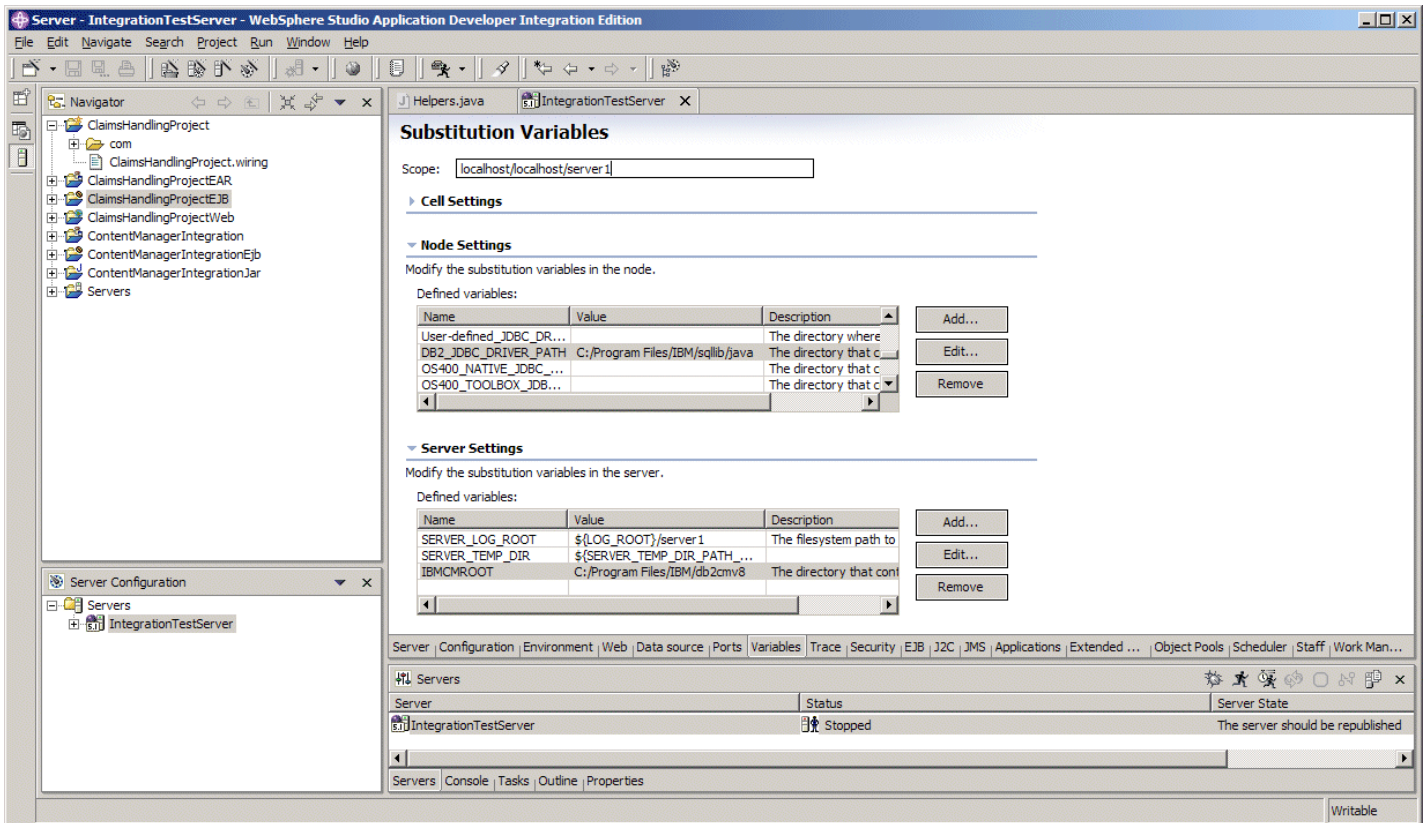


- Close the System Administration client.

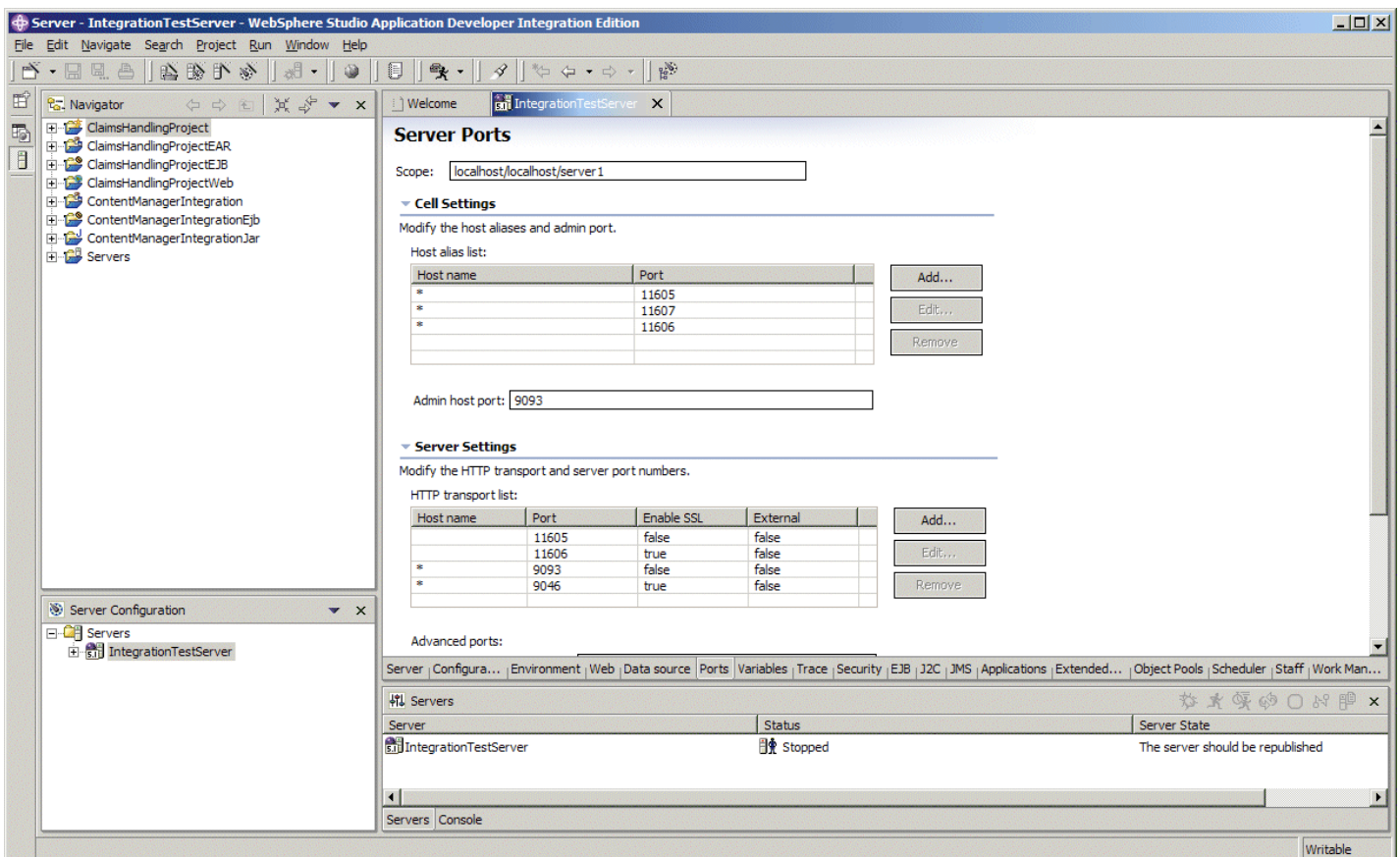
2.6 Configuring and Starting the Test Server

The Server project imported with the previous step configures a test server that can be used to run the sample process within *WebSphere Studio Application Developer Integration Edition*. The test server configuration points to the *WebSphere Business Integration Server Foundation* located in `<WSADIE_HOME>\runtimes\ee_v51` (note that it was formerly named Enterprise Edition). This step adjusts the environment and security settings of the test server configuration to the settings of your Content Manager Server.

- In *WebSphere Studio Application Developer Integration Edition* switch to the **Server** perspective by clicking **Window > Open Perspective > Server** or using the shortcut in the toolbar.
- In the Server Configuration view, expand **Servers** and double-click **IntegrationTestServer** to open the Server Configuration editor.
- Click the **Variables** tab and verify that the following two variables point to the appropriate locations in your file system:
 - In the section **Node settings** ensure that `DB2_JDBC_DRIVER_PATH` points to `<DB2HOME>\java`
 - In the section **Server settings** ensure that `IBMCROOT` points to `<IBMCROOT>`

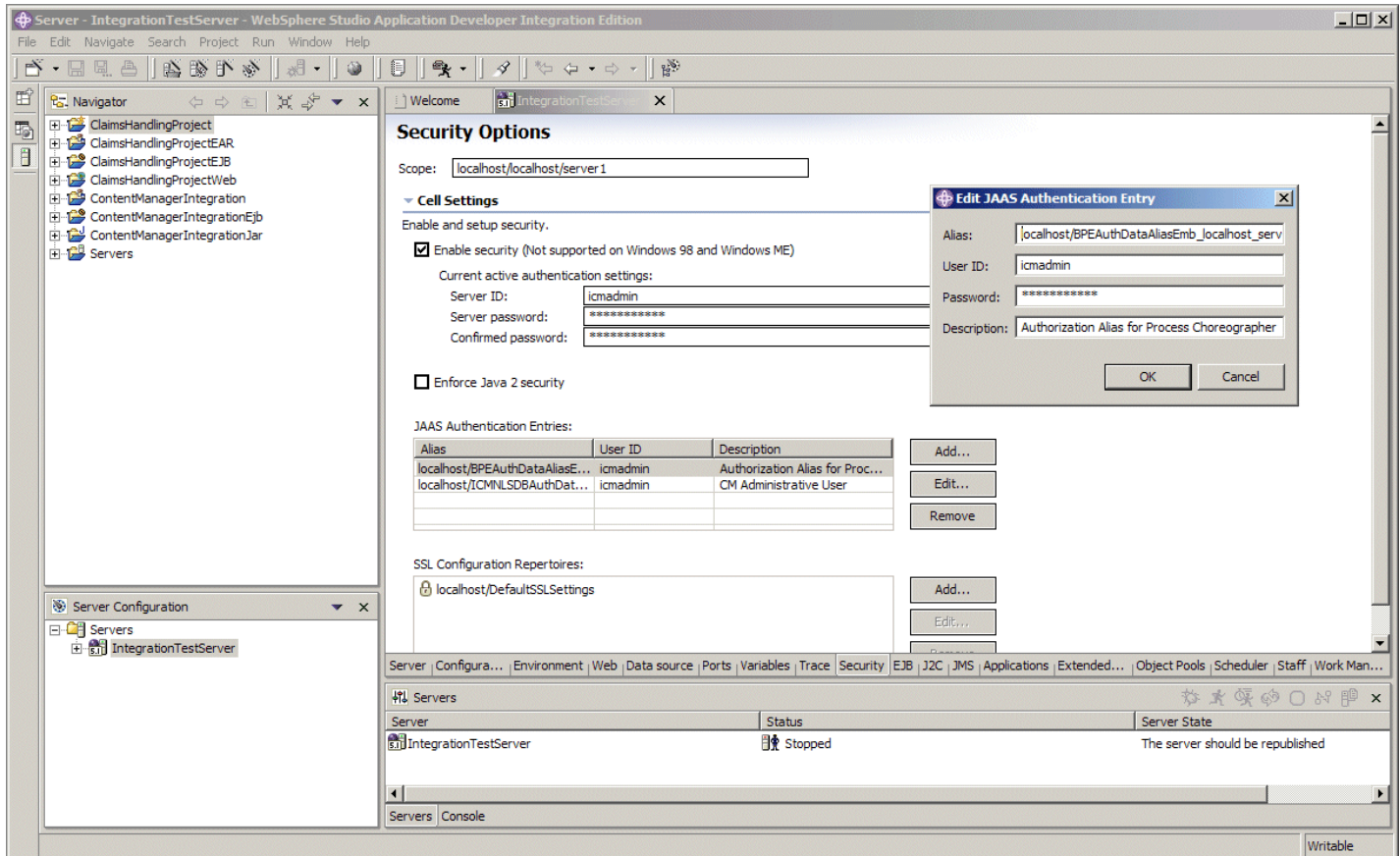


- Click the **Ports** tab and ensure that the ports used by the test server (9046, 9093, 11605...11607) do not conflict with port settings on your system. Change the ports in case of a conflict. Note that in this case the link that opens the initial page of the Quick Start Client shown in the next section might not work since it assumes the port number is 11605.

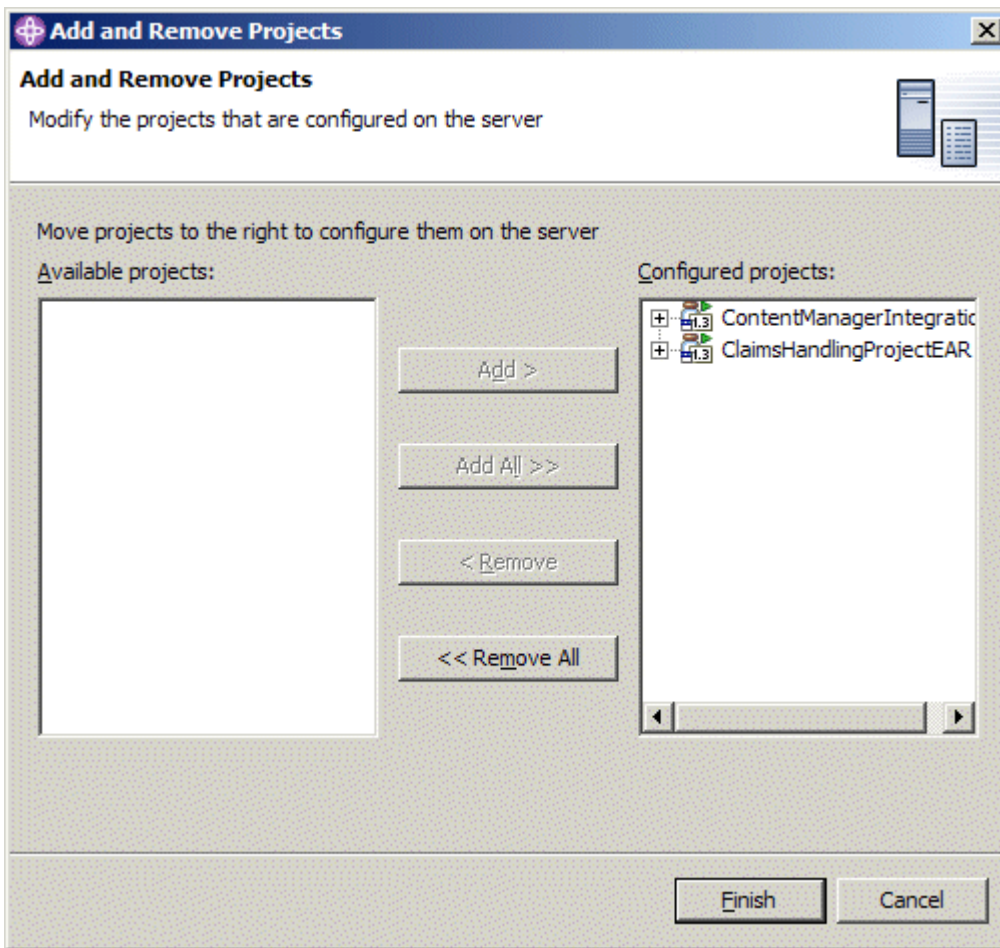


- Click the **Security** tab and specify the following authentication settings:
 - In the **Current Active Authentication Settings** section enter the userid/password of the CM library server administrator (icmadmin) into the Server ID/Server password fields

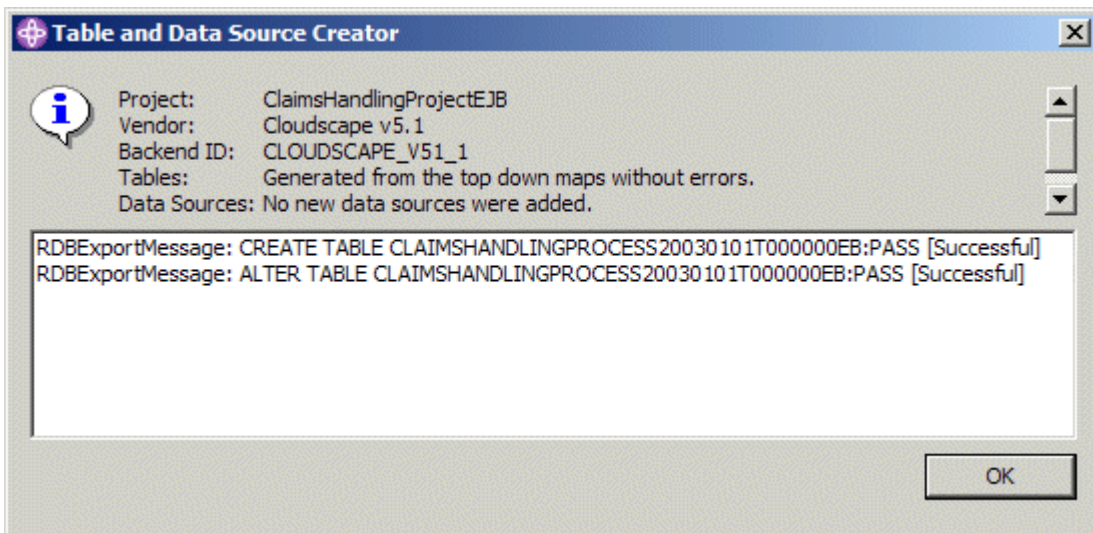
- In the **JAAS Authentication Entries** section click **edit** for the two fields localhost/BPEAuthDataAliasEmb_localhost_server1 and localhost/ICMNLSDBAuthDataAlias and enter the userid/password of the CM library server administrator



- Save your changes by pressing CTRL-S and close the editor for the IntegrationTestServer, because this editor must be closed before deploying projects later on.
- Right-click the Server Configuration **IntegrationTestServer** in the Server Configuration view and select **Add and remove projects**. On the **Add and Remove Projects** window select **Add All >>** and click **Finish**

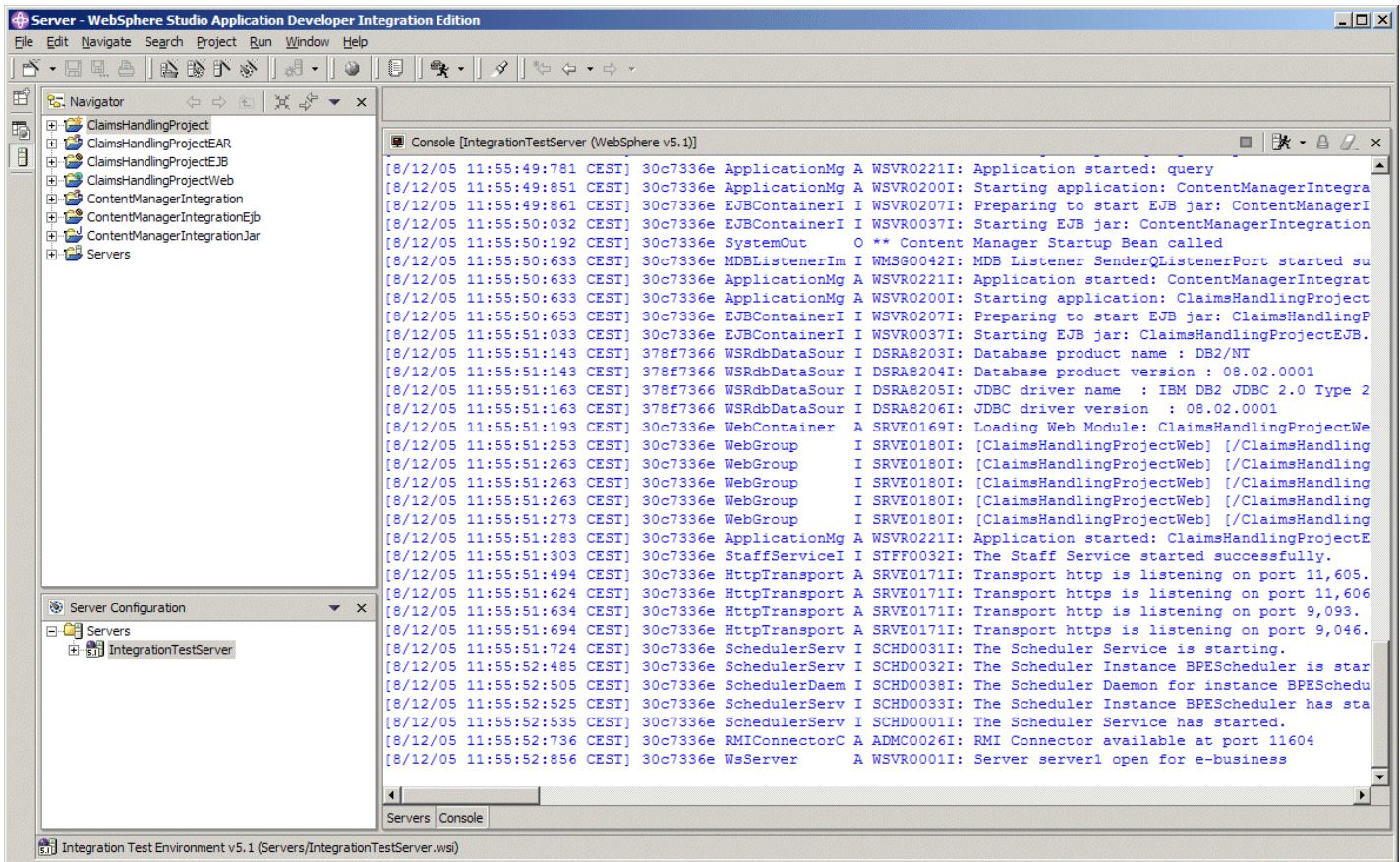


- Right-click the Server Configuration **IntegrationTestServer** in the Server Configuration view and select **Create Tables and Data Sources**. The result window looks as follows:



Click **OK** to close this window.

- In the Server view select the **Servers** tab, right click **IntegrationTestServer** and select **Start**. This publishes the project definitions to the server and then starts it.
- Wait until the message "Server server1 is ready for e-business" shows up in the console tab. The end of the console output should look similar to the one shown below:



Note that the output says `** Content Manager startup bean called` which indicates that the initialization step of the integration component has been passed successfully.

3. A Tour of the Sample Process

A note on the new sample process: the integration sample process provided with *DB2 Content Manager Enterprise Edition V8.3* was designed to be compatible with the sample process used to illustrate **document routing** (*DB2 Content Manager Enterprise Edition V8.3* includes a document routing engine for use with document lifecycle management within Content Manager).

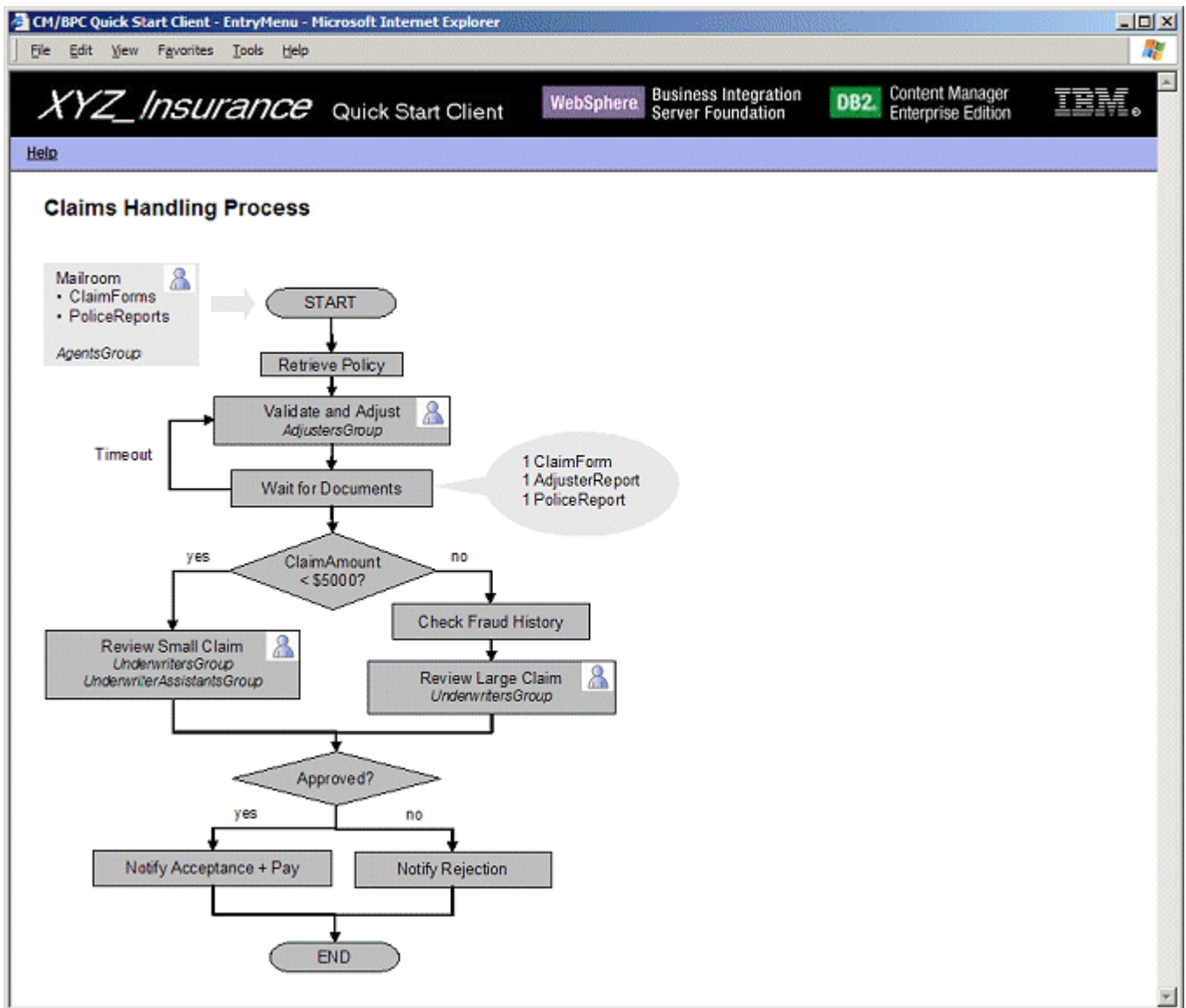
The new sample process provided with CM / PC Integration Quick Start focuses on the integration of different services both human-based and automatic. Automated services such as retrieval of the insurance policy or the fraud history check illustrate the inclusion of third party applications or B2B transactions with external service providers.

This is a more natural way of demonstrating the use of Process Choreographer since the Content Manager Enterprise Edition V8.3 sample process is a simple human-centric document flow example that can easily be implemented on the basis of Document Routing.

3.1 The sample process

The sample process provided with the CM / PC Integration Quick Start implements a simplified Claims Handling Process. Though it uses some insurance terminology and corresponding sample data it can be seen as an example of a more abstract review and approval process.

- Click this link: <http://localhost:11605/QuickStartClient> to display the initial page of the Quick Start Client. This page shows an abstract representation of the process outlining important steps and roles. You may want to bookmark this link so you can easily launch the client without reference to this documentation. Note that you have to adapt the port number in the link to the Quick Start Client if you have changed it during configuration (see [Configure and Start the Test Server](#))



- The blocks that have a schematic representation of a person in their top right corner represent staff activities. The other blocks represent (automated) services. Diamonds represent decisions and the rounded rectangles indicate the beginning and end of a process instance's lifetime (the process is configured such that process instances are deleted as soon as they terminate).
- You can move the cursor over the graphical elements to see a brief description of their purpose.
- The **Mailroom** represents an activity that is outside the scope of the process. This activity is responsible for adding incoming documents to the Content Management System. Note that the functions of the Mailroom could also be performed with a Content Manager client such as the *DB2 Content Manager Client for Windows*.
- The process enclosed by rounded rectangles consists of three sections: 1. data validation, 2. approval or rejection, 3. notification.

1. **Data validation** ensures all information needed to review the case is available. It consists of an automated step (RetrievePolicy), a staff-based one (Validate and Adjust) and a so-called **Collection Point**. Retrieve Policy uses the **Folder Management** service provided by the Integration Toolkit to locate the insurance policy for the claim. The staff-based activity Validate and Adjust assumes that additional information may be required as input for the review such as an Adjuster Report. Members of the AdjustersGroup are responsible for pre-assessing the case and making sure all required information that is needed to decide on the case is available. The collection point is a service provided by the Integration Toolkit that lets a process instance wait until a certain number of specified items is in the claim folder. The members of the AdjustersGroup can modify these conditions to a certain degree (such as removing the dependency on a Police Report). The collection point service performs regular checks on the condition and lets the process instance continue if all required documents are in the folder. A timeout occurs if the required documents are not available in a specified time frame. In this case process execution loops back to the Validate and Adjust activity where a reminder count keeps track of the number of timeout cycles that happened so far. At this activity staff personnel can act accordingly by requesting the required documents or weakening the collection point condition. See the section [The collection point service](#) for details on the collection point service.

2. The **Approval or rejection** section of the process begins with a switch activity (also called a decision point) where one of two paths is taken depending on the value of the **Claim Amount** attribute. For small claims members of the UnderwritersGroup investigate the documents in the folder and complete the review activity by approving or rejecting the claim. For large claims (Claim Amount > \$5000) a **FraudHistory** service assesses the trustworthiness of the claimant and might check certain criteria of the case. The result of this service is made available to members of the UnderwritersGroup who are entitled to review either small or large claims. Only members of the UnderwritersGroup can approve or reject large claims.
3. The **Notification** section of the process evaluates the result of the review and either sends an approval letter and pays the claim amount or sends a rejection letter. Note that sending of the approval letter and payment (which we assume to be a sub-process) can run in parallel. The CheckFraudHistory service and the three services involved in the notification section are implemented as simple Java methods that write a message to the server's standard output since illustrate the use of internal or external services but are beyond the scope of the sample. These methods can be found in <WORKSPACE_FOLDER>\ClaimsHandlingProject\com\ibm\bpe\cm\sample\ClaimsHandlingUtilities.java.

- Moving the cursor over the boxes of the process diagram displays a brief description of the purpose of the corresponding element.
- To perform a staff-based activity, click on a box that represents a human activity and log on as a member of the group displayed in italics right below the box title. The following table shows the mapping of actions to group names and group members:

| | Group | Group members |
|-------------------|----------------------------|--|
| Mailroom | AgentsGroup | Agent1, Agent2 |
| ValidateAndAdjust | AdjustersGroup | Adjuster1, Adjuster2 |
| ReviewSmallClaim | UnderwriterAssistantsGroup | UnderwriterAssistant1, UnderwriterAssistant2 |
| ReviewLargeClaim | UnderwritersGroup | Underwriter1, Underwriter2 |

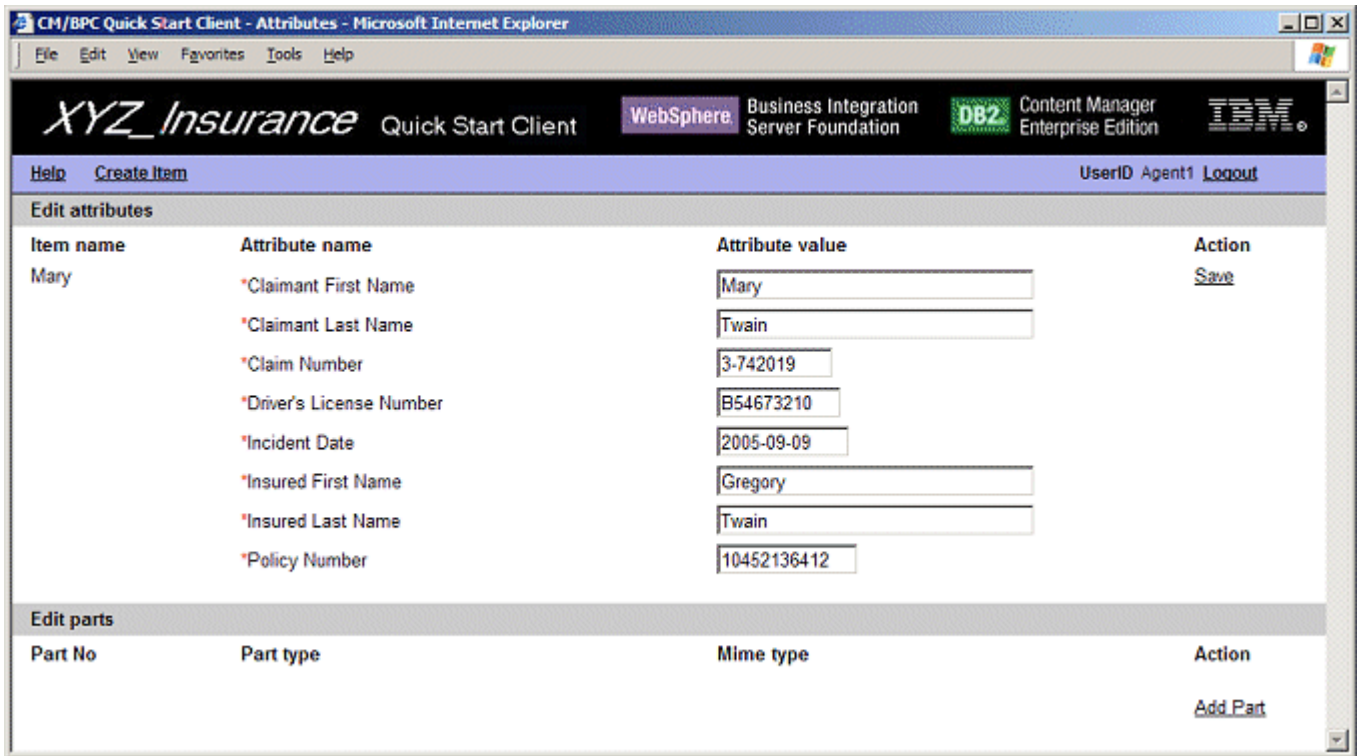
If your library server name is not `icmnlbdb` you need to open <WORKSPACE_FOLDER>\QuickStartClient\JavaSource\com\ibm\bpe\cm\util\ProcessData.java in *WebSphere Application Developer Integration Edition*, change the value of `CM_DATASTORE_NAME` accordingly, and re-build the project `QuickStartClient`. Note that the update is enabled in the active Web project on the fly so the server does not need to be re-started.

3.2 Process initiation

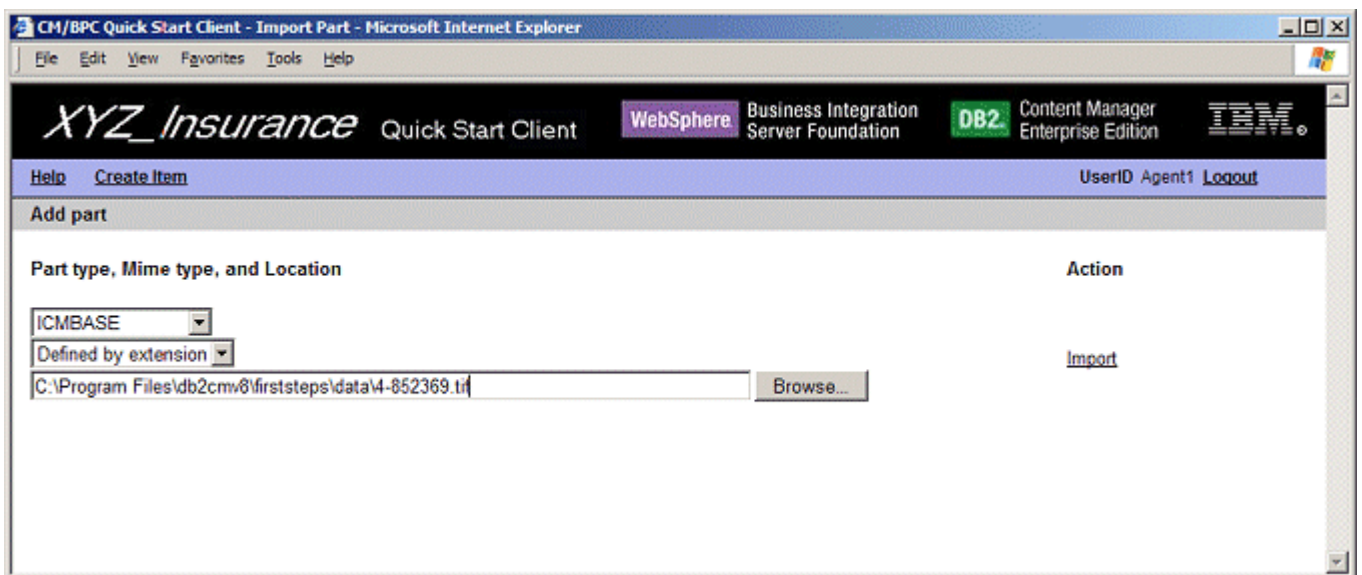
This section explains the different ways to create process instances corresponding to a variety of options to import documents into the Content Manager server. The key distinction is if process instances are created explicitly by an application or implicitly based on events. The former option may be preferred if the application that manages incoming documents is within the control of your organization so that code can be added to explicitly create process instances where needed. The code used to create instances based on existing items may be investigated to see how this works. The latter option may be preferred in cases where document import is treated as a black box or the corresponding application is not within the control of your organization.

- Click the box titled **Mailroom** and log on as `Agent1` with password `passw0rd` (which by default is the same for all roles of the sample process).
- The Mailroom user's initial page contains three sections titled **Create documents**, **Import predefined documents**, and **Create process instances for existing items**

- The section titled **Create documents** can be used to create instance of the three item types XYZ_ClaimForm, XYZ_AdjReport, and XYZ_PolReport. Since each of these is linked to a XYZ_ClaimFolder that has the attribute WF_OnCreate with a default value of ClaimsHandlingProcess creating an instance of any of these creates an instance of an enclosing XYZ_ClaimFolder which is started on a ClaimsHandlingProcess instance.
- Since creating an item may involve entering many mandatory values the section **Import predefined documents** can be used to create items of the three types based on pre-defined attribute values. These values refer to the Content Manager first steps sample data and are explicitly stored in startProcessInstances.jsp. The **ClaimNumber** is displayed in an input field to enable simple creation of new cases. Note that the Claim Number is the attribute that uniquely defines a case. Set the value in the Claim Number appropriately to trigger the creation of a new claim folder or to ensure proper linkage of documents to an existing claim folder. For items of type Adjuster Report the claim amount can be specified to be either small (\$1000) or large (\$10000, \$50000) based on pre-defined values. Click **Store** to create a new item in Content Manager and store the predefined / modified attribute values. Note that the actual document files need to be added in a separate step by clicking **Add Part** on the 'edit attributes and parts' dialogue.
- The section **Create process instances for existing items** can be used to create process instances based on the sample data already imported into Content Manager during the 'Create Sample Data' step of the Content Manager First Steps application. Click **Submit** to run the default query /XYZ_ClaimFolder which locates all instances of XYZ_ClaimFolder and creates instances of the ClaimsHandlingProcess for each element of the result list. By adding additional conditions (as for example /XYZ_ClaimFolder/OUTBOUNDLINK[@LINKTYPE = "DKFolder"]/@TARGETITEMREF => XYZ_ClaimForm[@ClaimLName = "Twain"]) the query can be refined to restrict the items for which process instances should be created to e.g. all claim forms submitted by a person with family name Twain.
- This tour starts with the section **Import predefined documents** since this illustrates the use of content event handling and is based on existing data.
- Click the **Store** link for Mary Twain's claim form in the **Import predefined documents** section to create an instance of the XYZ_ClaimForm item type in Content Manager. Due to auto foldering this triggers the creation of a corresponding XYZ_ClaimFolder. Since the XYZ_ClaimFolder item type has the attribute WF_OnCreate defined and pointing to a default value of ClaimsHandlingProcess this triggers the creation of a process instance.
- The console output of the test server confirms the creation of an item (-> Item created) and completion of the first step in the process (>> Number of matching insurance policies: 1).
- Click **Add part** on Agent1's activity page to attach the actual document to the item.



- Locate the TIFF image of a scanned claim form in <IBMCROOT>\firststeps\data\4-852369.tif and click **Import** to add it to the new item. Note that the console output confirms the upload with a message (-> imported file of length: 27688 mime type: image/tiff)

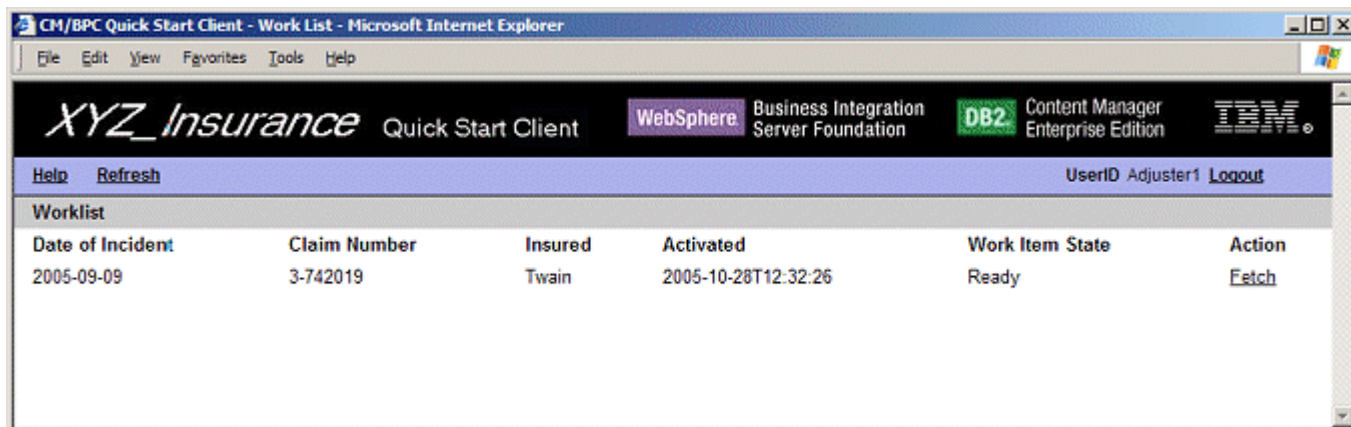


- You can click **Create item** to get back to the Mailroom home page and create further items. For now we log out Agent1 by clicking **Logout** at the right hand side of the menu bar.

3.3 Data validation

- The message >> Number of matching insurance policies: 1 in the server console confirms that an appropriate insurance policy has been located in content manager which has the policy number as entered in the claim form. The RetrievePolicy service has retrieved this item from Content Manager and added it to the folder that is routed through the process.
- On the process diagram click the box **Validate and Adjust** and log on as Adjuster1.
- The adjuster's initial page shows the adjuster's worklist. It contains a single work item corresponding to the process instance that has been created in the previous step (Mary Twain's claim form). The worklist might be empty due to the process being in the 'Retrieve Policy' state. In this case the **Refresh** may need to be clicked to update the worklist to its

most recent state. Note that the worklist display results from a combined query that returns item attributes (Date of Incident, Claim Number, etc.) as well as process properties (Activated, Work Item State). The console output shows the Content Manager query in the form `query= /XYZ_ClaimFolder[@ITEMID IN]`



- Click **Fetch** to check out this work item. This makes it unavailable for any other member of the AdjustersGroup and brings up the work item view. The console output confirms that the work item has been checked out with the message -> Work item claimed

CM/BPC Quick Start Client - Work - Microsoft Internet Explorer

File Edit View Favorites Tools Help

XYZ Insurance Quick Start Client WebSphere Business Integration Server Foundation DB2 Content Manager Enterprise Edition IBM

Help My ToDos UserID Adjuster1 Logout

Process activity

ValidateAndAdjust

Contract retrieved Required documents Adjuster Report 1 Remind me in 10 seconds

Reminder count:0 Police Report 1

Auto Photo 0

Actions Complete Cancel

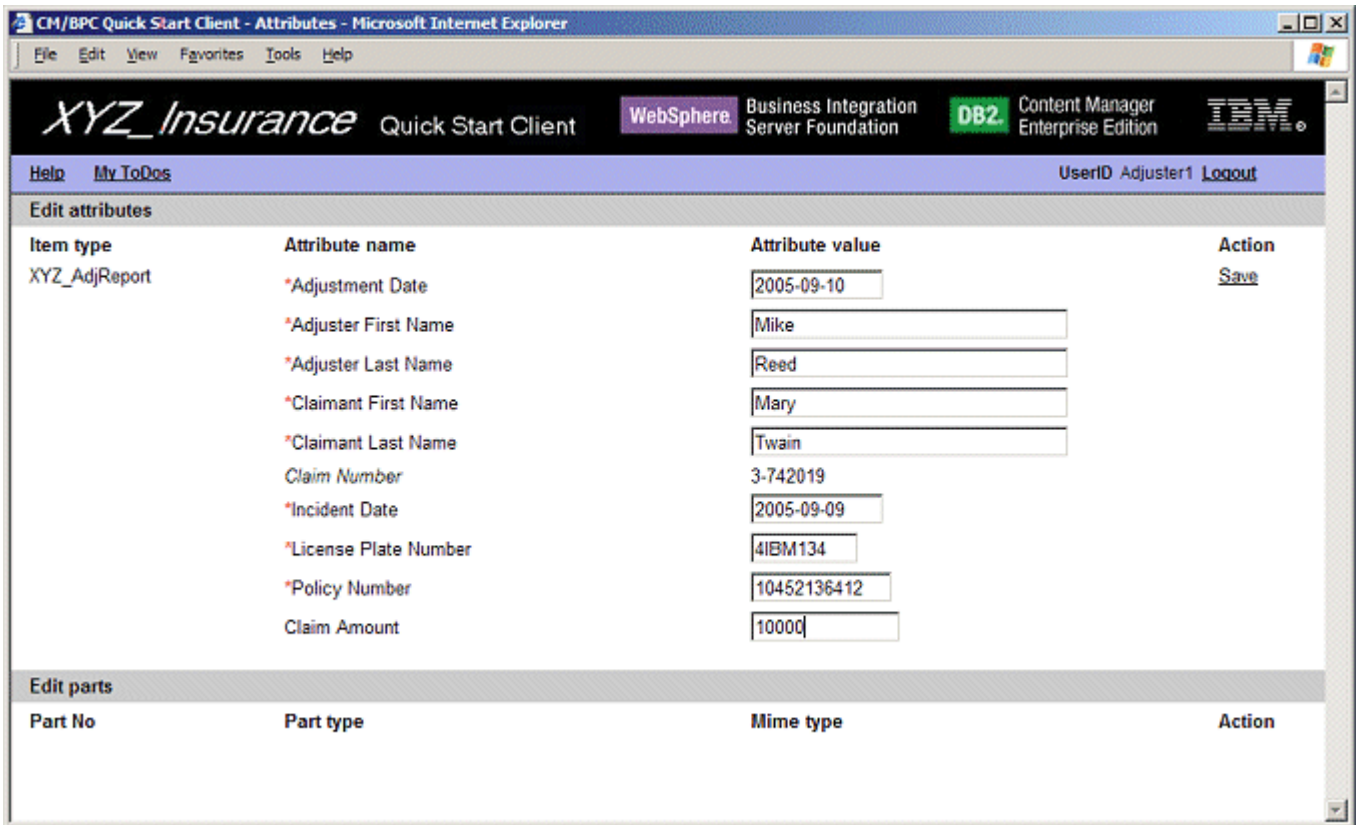
Folder

| Folder Name | Type / Parts | Attribute name | Attribute value | Actions |
|-------------|--------------------------|----------------|-----------------|------------------------------------|
| 3-742019 | Claim Application Folder | Claim Number | 3-742019 | Edit Add Adjuster Report Add Photo |

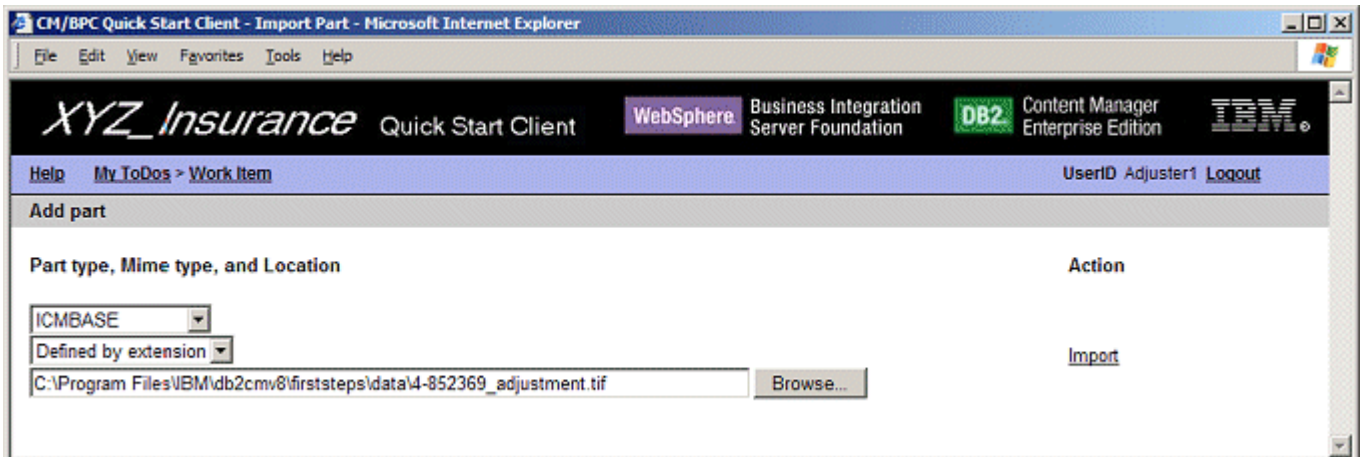
Items in folder

| Item Name | Type / Parts | Attribute name | Attribute value | Actions |
|-----------|-----------------|-------------------------------|-------------------|-------------|
| Mary | Auto Claim Form | Claimant First Name | Mary | Edit Delete |
| | | Claimant Last Name | Twain | |
| | | Claim Number | 3-742019 | |
| | | Driver's License Number | B54673210 | |
| | | Incident Date | 2005-09-09 | |
| | | Insured First Name | Gregory | |
| | | Insured Last Name | Twain | |
| | | Policy Number | 10452136412 | |
| | | 10452136412 | Insurance Policy | |
| | | Street | 258 Green Rd. | |
| | | State | CA | |
| | | City | Ocean | |
| | | ZIP Code | 90060 | |
| | | XYZ_Insured | | |
| | | Insured First Name | Gregory | |
| | | Insured Last Name | Twain | |
| | | XYZ_Insured | | |
| | | Insured First Name | Mary | |
| | | Insured Last Name | Twain | |
| | | XYZ_VIN | | |
| | | Vehicle Identification Number | 1HOME10R4KL987258 | |
| | | XYZ_VIN | | |
| | | Vehicle Identification Number | 3CASA55R4KL987500 | |
| | | XYZ_VIN | | |
| | | Vehicle Identification Number | blurx | |
| | Part 0 | ICMBASE | image/tiff | View |

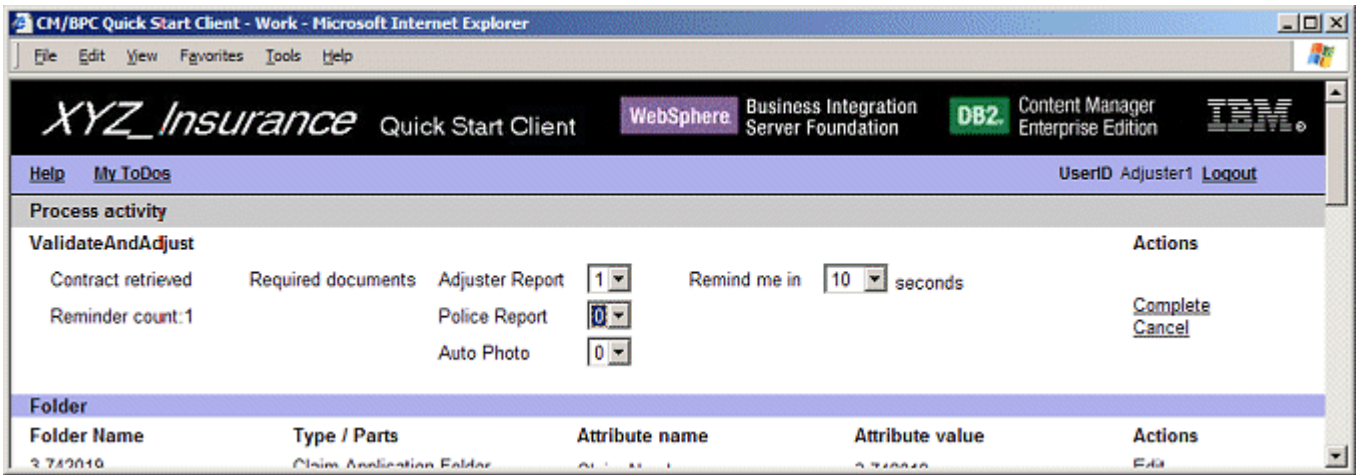
- Note that the work item page contains three sections titled **Process activity**, **Folder**, and **Items in Folder**. The **Process activity** section basically displays the in- and output message of the staff activity that consists of the relevant process properties (input message) and of the controls which the Adjuster needs to work with to complete this step of the process. The **Folder** and **Items in folder** section display the content-related information. The first column shows the folder or item name as specified by the 'represents item' property of the corresponding item type. The second column shows the item type or part number and the third/fourth column display attribute value pairs of the corresponding item or folder.
- Optional: at this point you may want to explore how a user would modify attribute values. You can do so by clicking **Edit** and for example change the value of the *Street* field in the address to *259 Blue Rd.*. When clicking **Save** you will see the effect of the modification on the updated work item page.
- Click **View** to open the viewer in a separate window. Create an annotation (you may highlight a part of the document or put a note on it) which may be used to communicate important information to a human role (e.g. Underwriter) later in the process.
- Note that the Insurance Policy has been retrieved and added to the folder by the first activity of the process.
- Click the **Add Report** action in the folder section to add an item of type Adjuster Report. This opens the **Edit attributes and parts** page. Note that the claim number is already instantiated to the claim number of the folder. Enter values for the mandatory fields (marked by a red asterisk) as shown below and click **Save** to store the adjuster report in Content Manager.



- Click **Add part** and import <IBMCROOT>\firststeps\data\4-852369_adjustment.tif. Click **Work Item** in the menu bar to return to the **Work Item View** and note that the Adjuster Report is now a member of the claim folder.



- Note that the collection point conditions require an additional Police Report before the process can continue. Click **Complete** in the 'Process activity' section to move the folder to the next step in the process to experience the effect of a collection point timeout.
- After 10 seconds (depending on the 'Remind me' setting) the following message shows up in the console view: Reminder: submit remaining documents for claim number 3-742019 iteration (1). Click Refresh in the menu bar of the Quick Start Client to refresh the adjuster's worklist. Notice that it again contains an entry referring to case 3-742019.
- Notice that the 'Reminder count' now has the value 1 since this is the first time the process has cycled back to the ValidateAndAdjust activity due to the collection point conditions not being met.
- In the 'Process activity' section set the number of required documents of type Police Report to 0 assuming that the police report is not required for a decision later in the process. This shows how collection point conditions can be set dynamically on a process instance.



- Click **Complete** to complete the adjuster's activity and forward the work item to the next step in the process. The console output displays the message -> Work item completed to confirm successful completion of this step.
- Since Adjuster1 has changed the number of required Police Reports to 0 the collection point condition is now met by the folder since it contains the three required documents: a ClaimForm, an Insurance Policy, and an Adjuster Report. The console output confirms that the collection point has been passed successfully with the message >> Documents are ready for claim number 3-742019. It further displays the following two messages: >> Claim Amount is large (10000.00) and >> Fraud report for Mary Twain is good that result from the decision point and the Check Fraud History service respectively.
- Log out Adjuster1.

3.4 Approval or rejection

- Since the adjuster report created in the previous step lists a claim amount of \$10000 the claim is considered a large claim by the decision at the beginning of this section which moves it along the path with the two activities **CheckFraudHistory** and **ReviewLargeClaim**.
- The **CheckFraudHistory** service is a service that decides based on some data extracted from the relevant documents if the claimant's fraud history is good or bad. This could be realized by a BI tool that investigates certain company-internal data bases such as statistics about incident locations, customer credibility, etc. Alternatively it could be an external service provided by some sort of agency with which the insurance company shares a contract. The sample implements this as a Java-based service that returns a bad fraud history if the initial letter of the claimant is in the range 'A'...'L' and returns good otherwise.
- On the process diagram click the box **Review Large Claim** and log on as Underwriter1.
- The home page of the underwriter activity shows the Underwriter's worklist. It contains a single work item corresponding to case number 3-742019. Note that the columns in the work list differ from the adjuster's worklist in that they now contain the name of the current activity (members of the group UnderwritersGroup may perform either the ReviewSmallClaim or the ReviewLargeClaim activity) and the creation date of the adjuster report.



- Click **Fetch** to check out this work item and open the work item page

The screenshot displays the XYZ Insurance Quick Start Client interface. At the top, there's a navigation bar with 'Help' and 'My Todos'. The main content area is titled 'Process activity' and shows a 'ReviewLargeClaim' process. The process result is 'Fraud report for Mary Twain is good'. There are two radio buttons: 'Approve the claim.' (unselected) and 'Reject the claim.' (selected). A 'Justify decision:' text box is empty. To the right, there are 'Complete' and 'Cancel' links. Below the process activity, there are two tables. The first table, 'Folder', has columns for 'Folder Name', 'Type / Parts', 'Attribute name', 'Attribute value', and 'Actions'. It lists a folder '3-742019' of type 'Claim Application Folder' with attribute 'Claim Number' and value '3-742019'. The second table, 'Items in folder', lists items like 'Mary' (Auto Claim Form) and '2005-01-01' (Adjuster Report) with their respective attributes and actions like 'View', 'Edit', and 'Delete'.

- The **Process activity** section displays the result returned by the Check Fraud History service and the controls for approving or rejecting the claim and adding a justification. The justification is stored in a process variable and not in Content Manager. Thus it is available throughout the process and potentially stored in the process log but not made persistent in the content repository.
- Click **View** in the Actions column for the Auto Claim Form and investigate the annotations which the Adjuster has added to the document.
- Select **Approve** leave the justification field empty and click **Complete**.
- Notice the messages >> Paying for claim number 3-742019 and >> Sending approval letter for claim number 3-742019 that signal successful completion of the process.

3.5 Exploring alternative options

- Use the Mailroom activities to create adjuster reports and police reports. Ensure that the claim numbers of these reports are correct so auto foldering assigns them to the appropriate folders.
- Create process instances explicitly based on the claim folders in DB2 Content Manager (using the query option).

- Set Claim Amount to 1000 to explore the 'small claim' path. In this case log on as UnderwriterAssistant1.
- Reject a claim and notice the message >> Sending rejection letter for claim number etc...
- Note that the availability of operations in the right-most column of a staff activity (for example Edit, View, Delete) depends on the access rights members of this user group have for the corresponding item type. Remove rights to see the action list changes accordingly.
- Import parts with different mime types (note that when importing files with extension .jpg or .jpeg mime the mime type should be selected explicitly).
- Explore the capabilities of the viewer applet.

4. Developing a Custom Content Manager / Process Choreographer Integration Solution

This chapter describes the elements of the Integration Toolkit and outlines how they can be combined to create a custom integration solution.

4.1 How a CM/PC integration workspace is organized

The following sections assume a basic familiarity with *DB2 Content Manager Enterprise Edition V8.3*, *WebSphere Studio Application Developer Integration Edition V5.1.x*, and *WebSphere Business Integration Server Foundation V5.1.x*. They focus on the concepts that are specific to an integration solution based on the Integration Toolkit and Quick Start Client. Valuable resources to get familiar with the environment are the redbooks and documentation referenced in the literature section at the end of this document. From the point of view of business process creation we recommend the business process samples that can be found in *WebSphere Application Developer Integration Edition* under **New > Other > Examples > business integration > scenarios > BPEL**. We recommend working through at least one of the 'build it yourself' style samples and taking the time to look up new concepts in the redbooks or documentation.

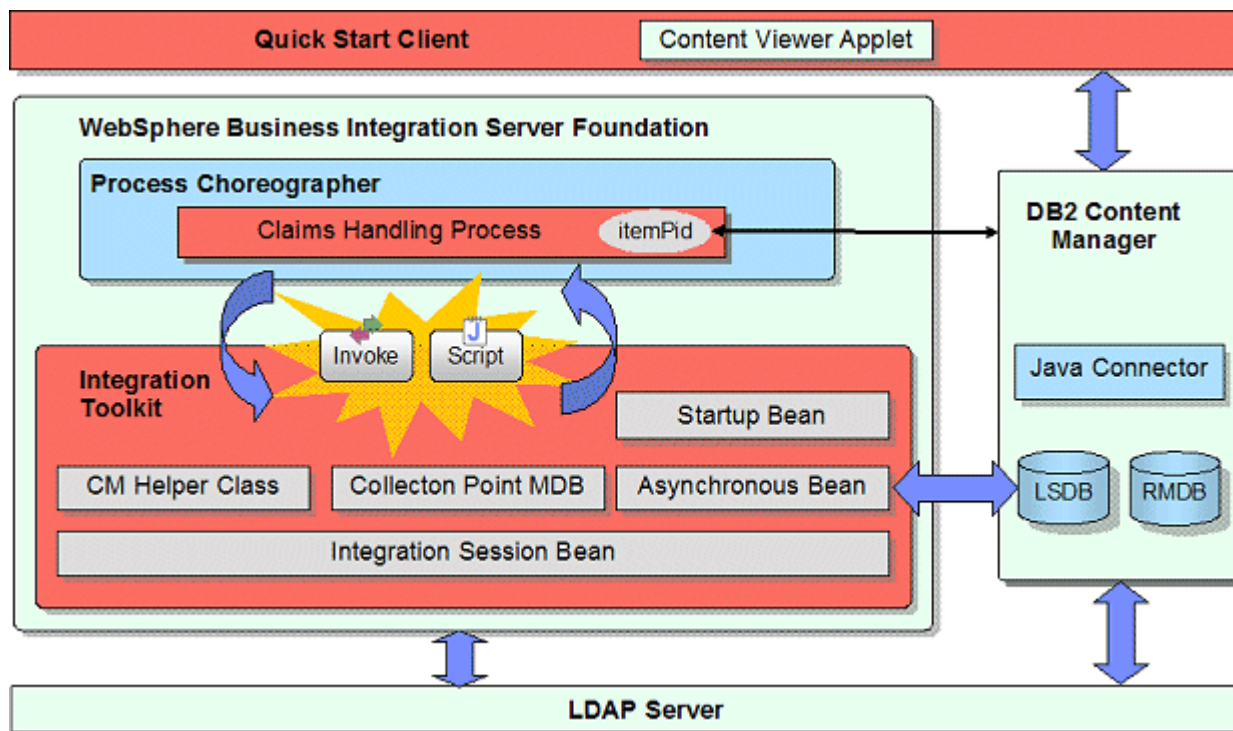
A workspace of a custom solution that integrates DB2 Content Manager with Process Choreographer typically consists of the following projects:

- Service project(s) that host(s) one or more processes. A useful convention may be to use a name ending with 'Process' to distinguish it from the other projects in the workspace.
- Integration Toolkit (ContentManagerIntegration, ContentManagerIntegrationJar, ContentManagerIntegrationEjb).
- Quick Start Client adjusted to the needs of the custom process.

In the following we describe the elements of an integration solution based on the three types of projects and their interaction.

4.2 The Integration Toolkit

The Integration Toolkit is a set of Java classes and EJBs that support the creation of content-centric processes. See the figure below for a schematic representation of the Quick Start architecture:



The three red blocks mark the three elements of the Content Manager / Process Choreographer Integration Quick Start: the Quick Start Client, the claims handling sample process and the Integration Toolkit.

The Integration Toolkit methods are implemented within a stateless session bean context that permits the current global transaction to be suspended while invoking the Content Manager API. This enables efficient use of Content Manager within the WebSphere Application Server environment. A Java class named `ContentManager` provides static methods that simplify programmatic exploitation of underlying services, including EJB access. This class is especially useful in the Process Choreographer Java Snippet environment, where process designers may have limited Java skills.

The API documentation for the static methods in `ContentManager` can be found in `ContentManagerIntegrationJar\doc\index.html`

Startup beans are WebSphere-specific stateful session beans that are automatically notified when a WebSphere application is started or stopped. A bean is designated as a startup bean by implementing special home and remote interfaces. When WebSphere starts an application, it looks for beans that implement these special startup bean interfaces. If it finds any, it arranges to invoke the beans `start()` and `stop()` methods on application initialization and termination. The Integration Toolkit uses the startup bean to start and stop the asynchronous bean described below.

Asynchronous beans are WebSphere-specific enterprise beans that are executed asynchronously and provide support for application threading within a J2EE environment. There are three types of asynchronous beans:

- Worker Beans
- Alarm Listener Beans
- Event Listener Beans

The Integration Toolkit uses a worker asynchronous bean. A worker asynchronous bean is like a Windows service or a Unix/Linux daemon process. It runs until asked to terminate. The integration asynchronous bean runs in a loop, doing its work and then sleeping for a period of time before waking up and repeating the process. When started by the startup bean, the asynchronous bean used by the Integration Toolkit invokes methods in the Content Manager Integration Stateless Session Bean to perform the following operations:

- Check for satisfied collection point criteria
- Process content manager events that require process initiation or termination actions (content event handling)

The asynchronous bean loops until its stop method is invoked by the startup bean at application termination.

The **Collection Point MDB** (Message Driven Bean) is invoked in response to messages from Collection Point activities in content-centric processes. In other words, when such a process encounters an Invoke activity which is bound to the Collection

Point service, it puts a message on a specific Java Message Service (JMS) queue (CollectionPointQueue). When that message reaches the head of the queue, an associated listener gets the message from the queue and WebSphere invokes the Collection Point MDB. The Collection Point MDB retrieves the elements of the request (item id, name and quantity of items to wait for) and invokes a method in the Content Manager Integration Stateless Session Bean to register the collection point for subsequent processing by the Content Manager Integration Asynchronous Bean, which sends a reply message back to the business process when the collection point condition is satisfied.

You can invoke an asynchronous service (e.g. bound to a JMS implementation) with a single invoke activity or use a invoke/receive pair in the BPEL process to communicate asynchronously with a partner. In the first case a correlation set needs to be maintained to locate the process instance to which the response should be sent. With JMS binding locating the appropriate process instance can more easily be done based on the JMS message ID.

The **Content Manager Integration Stateless Session Bean** does most of the work. Note that the implementation of most of these functions can be found in the `com.ibm.bpe.cm.util` package of the `ContentManagerIntegrationJar` project.

At the level of content / workflow interaction patterns **the Integration Toolkit offers the following functions:**

- *Content Event Handling* - item creation or deletion events in Content Manager trigger corresponding actions in Process Choreographer
- *Collection Point* - let a process instance wait until a certain number of items of a certain type are in a folder (or a timeout or exception occurs)
- *Content Attribute Access* - efficient means to access the value of content attributes from within the business process, e.g. to enable content-specific decisions
- *Folder Service* - add the item routed through the flow to a retrieved folder or add a retrieved item to the folder routed through the flow
- *Combined Query* - efficiently retrieve process information from Process Choreographer and associated business attributes from Content Manager
- *Common Staff Repository* - the Integration Toolkit includes a WebSphere User Registry that leverages the Process Choreographer staff resolution facility to map Content Manager user and group definitions to the role / verb model used to identify user access for a work item

A process that uses any of the toolkit functions needs to comply with the following conventions:

- It has a single receive activity.
- The input message definition (WSDL) of the process contains a part with the name `itemPid` and type `(xsd:string)`.
- The receive activity is followed by a Java snippet which performs the following initialization:

```
ContentManager.initializeCustomProperties(this, get<ProcessRequestVariableName>  
( ).getItemPid());
```

When creating a new CM/PC integration project make sure that `ContentManagerIntegrationJar` is included in the list of project references of the corresponding service project. The initial Java snippet copies the `itemPid` value of the input message to process properties where they are available e.g. for combined query evaluation. The Integration Toolkit stores an item's unique identifier as both a fully specified `itemPid` and as a triple `<ITEMID, COMPONENTID, VERSIONID>`. When starting a process instance based on content event handling, only the triple-based information is available. During initialization of the process, the method `initializeCustomProperties()` retrieves the fully specified item PID that contains among other information the library server name and item type.

| |
|---------------------------|
| SubmitClaimRequest |
| CollectionPointRequest |
| CollectionPointResponse |
| CollectionPointStatus |
| CheckFraudHistoryRequest |
| CheckFraudHistoryResponse |

```

graph TD
    SubmitClaim[SubmitClaim] --> InitializeProcessProperties[InitializeProcessProperties]
    InitializeProcessProperties --> AddPolicyToFolder[AddPolicyToFolder]
  
```

InitializeProcessProperties

Description

Implementation

Documentation

Join Behaviour

Server

```

// Store the itemId and its constituents in custom properties of the process instance

String itemId = getSubmitClaimRequest().getItemId();
ContentManager.initializeCustomProperties(this, itemId);

```

4.3 Content Event Handling

With content event handling, activities in DB2 Content Manager generate events that can be handled by the business process management system. The Integration Toolkit provides an implementation of content event handling that deals with item creation and deletion events. If defined properly, an item creation event triggers the creation of a process instance as soon as an item of a certain type is stored in DB2 Content Manager. The newly created process instance contains a reference to the item just created. If capturing of deletion events is specified, the deletion of an item in DB2 Content Manager triggers the removal of any process instances 'carrying' it.

How to use Content Event Handling

The administrator specifies which items trigger creation or deletion events based on the custom attributes `WF_OnCreate` and `WF_OnDelete` that need to be assigned to the item types for which content event handling should be enabled. Both attributes need to be defined as a variable length character string with a minimum length of zero and a maximum length of 254 as shown below:

Attribute Properties - WF_OnCreate [X]

Name: * WF_OnCreate

Display name: * WF_OnCreate Translate...

Attribute type

- Char_acter
- Var_iable character
- Short integer
- Long integer
- Dec_imal
- D_ouble
- Date
- Time
- Time_s_tamp
- BLOB
- CLOB

Character type

- Alphanumeric (1)
- Numeric (2)
- Alphanumeric (3)
- Extended alphanumeric (4)
- Other (5)

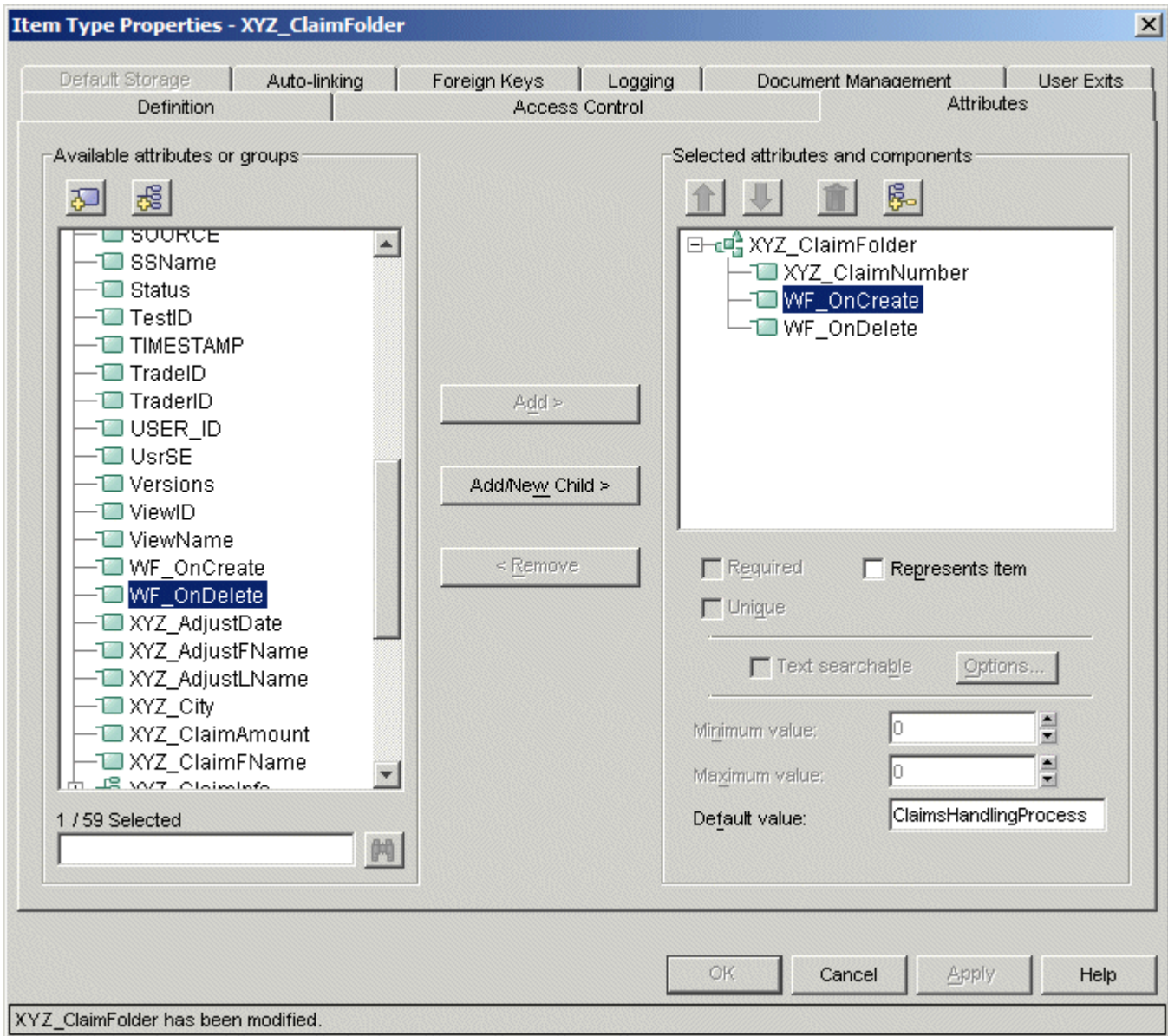
Character length

Minimum: 0

Maximum: 254

OK Cancel Apply Help

To enable an item type for content event handling add WF_OnCreate or WF_OnDelete to its list of attributes and enter the template name of the process to be notified as the default value.



The value of the attribute `WF_OnCreate` specifies the name of a process to start whenever the corresponding event occurs. So, for example, if the `WF_OnCreate` attribute for the `XYZ_ClaimFolder` item type is set to `ClaimsHandlingProcess`, then whenever a user creates a new item of type `XYZ_ClaimFolder`, a new instance of the `ClaimsHandlingProcess` is started with a reference to this item in its input message. Note that this assumes a certain definition of the input message for these processes, in particular the message must contain at least a String part named `itemPid`. If multiple versions of this process are deployed, the most recent process with respect to the **Valid From** date is selected. If the attribute `WF_OnDelete` is present in the definition of the item type `XYZ_ClaimFolder` with a default value of `ClaimsHandlingProcess`, deleting an item of this type terminates all instances of the `ClaimsHandlingProcess` that carry this item.

With this create and delete linkage, content event handling enables some degree of referential integrity between the content repository and the process management system in the sense that document creation or deletion causes corresponding action in the process management system in order to keep the two systems synchronized.

How Content Event Handling is implemented

Item type definitions that have a `WF_OnCreate` or `WF_OnDelete` attribute are registered in a table `BPECONTENTEVENTS` that the Integration Toolkit creates in the library server database. Each row in the `BPECONTENTEVENTS` table constitutes a content event request. The method `checkContentEvent()` of the stateless session bean is called regularly from the asynchronous bean to see if a new process instance should be created or if process instances should be deleted. It scans the `BPECONTENTEVENTS` table in the order and performs the corresponding creation or deletion operations in the order in which requests have been added to the table deleting the rows as soon as the corresponding action has been performed successfully. If an action fails, an error message is written to the log the row remains in the table and the event is retried during the next cycle of the asynchronous bean. The `BPECONTENTEVENTS` table is created during server startup if it does not exist yet.

The `BPECONTENTEVENTS` table has the following structure:

| | |
|--------------------------|----------------------------|
| <code>CREATETS</code> | <code>TIMESTAMP</code> |
| <code>ITEMID</code> | <code>CHAR (26)</code> |
| <code>COMPONENTID</code> | <code>CHAR (18)</code> |
| <code>VERSIONID</code> | <code>SMALLINT</code> |
| <code>REQUEST</code> | <code>SMALLINT</code> |
| <code>PROCESSNAME</code> | <code>VARCHAR (254)</code> |

The `CREATETS` column is the creation date and time of the content event. It ensures that content events are processed in the order in which they are received. The `ITEMID`, `COMPONENTID` and `VERSIONID` columns are the key values of the corresponding itemPid the creation or deletion of which triggered the event. The `REQUEST` column indicates the type of CM event: 1 for creation, 2 for deletion. The `PROCESSNAME` column indicates the name of the process to be created in case of creation events. Rows are written to the `BPECONTENTEVENTS` table using database triggers. These triggers are automatically created by the Asynchronous Bean for Content Manager item types that contain the custom attributes `WF_OnCreate` or `WF_OnDelete`. The trigger is created on the underlying Content Manager Component Table (a table of the form `ICMUTnnnnnnsss`). Once the trigger is created, create and delete operations for the item type result in requests being written to the `BPECONTENTEVENTS` table. This has the benefit of decoupling the Content Manager transaction from the subsequent Check Content Event transaction so that the performance impact to the Content Manager transaction is minimal. The `BPECONTENTEVENTS` table is created (if it doesn't already exist) via the `ejbCreate` method, the first time the Content Manager Integration Stateless Session Bean is created.

Apart from this specific mechanism content event handling can also be viewed as specific instance of a more general concept that may be applied to other special processing, as needed.

Examples might be:

- Notify process instances if the item they are referring to is updated
- Trigger a 'review process' within a certain period of time after a document has been created
- Trigger a process if a document is moved to a different storage system
- etc.

4.4 The Collection Point Service

A collection point can be used where the item that is routed through the process is a folder. A collection point is an activity in the process which waits until a certain folder condition is valid. This condition is expressed in terms of `<ITEMTYPE, QUANTITY>` pairs. A set of such pairs specifies how many instances of the named item types need to be in the folder for the process instance to resume execution. `<1, XYZ_ClaimForm>`, `<2, XYZ_AdjReport>`, `<1, XYZ_PolReport>` is an example that lets the process instance wait until one claim form, two adjuster reports, and one police report are available in the claim folder that is routed through the process.

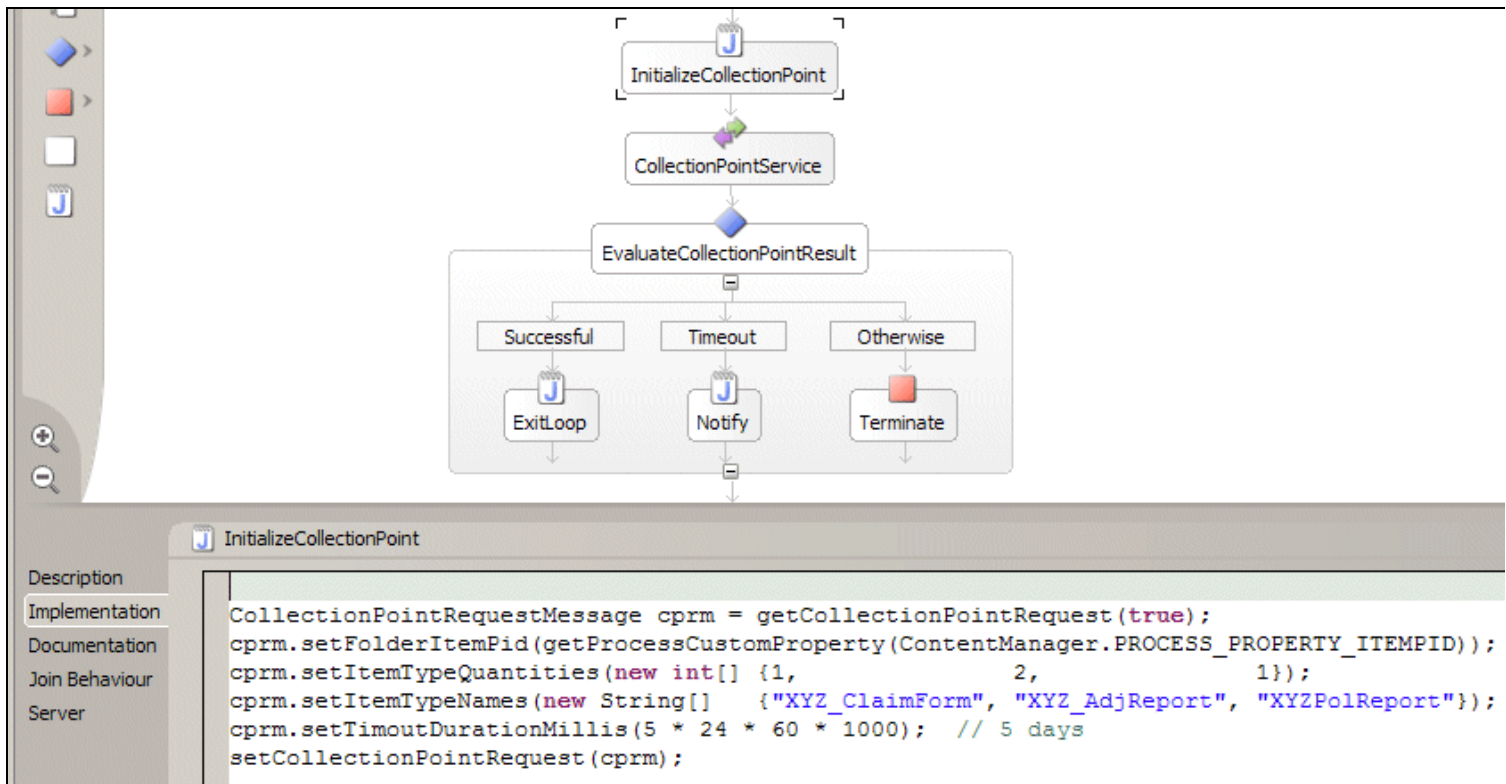
How to use the Collection Point Service

A collection point is modeled by two activities, a Java Snippet that defines the collection point conditions and an invoke activity that calls the collection point service provided by the Integration Toolkit. Note that a process using a collection point must be defined as long running since the availability of documents usually depends on some external conditions that are beyond the control of the process and typically person-based. Furthermore, a timeout may be specified to ensure the process instance resumes execution in lack of the required documents. In most cases, the timeout triggers a reminder or corrective action to ensure the process instance does not wait forever.

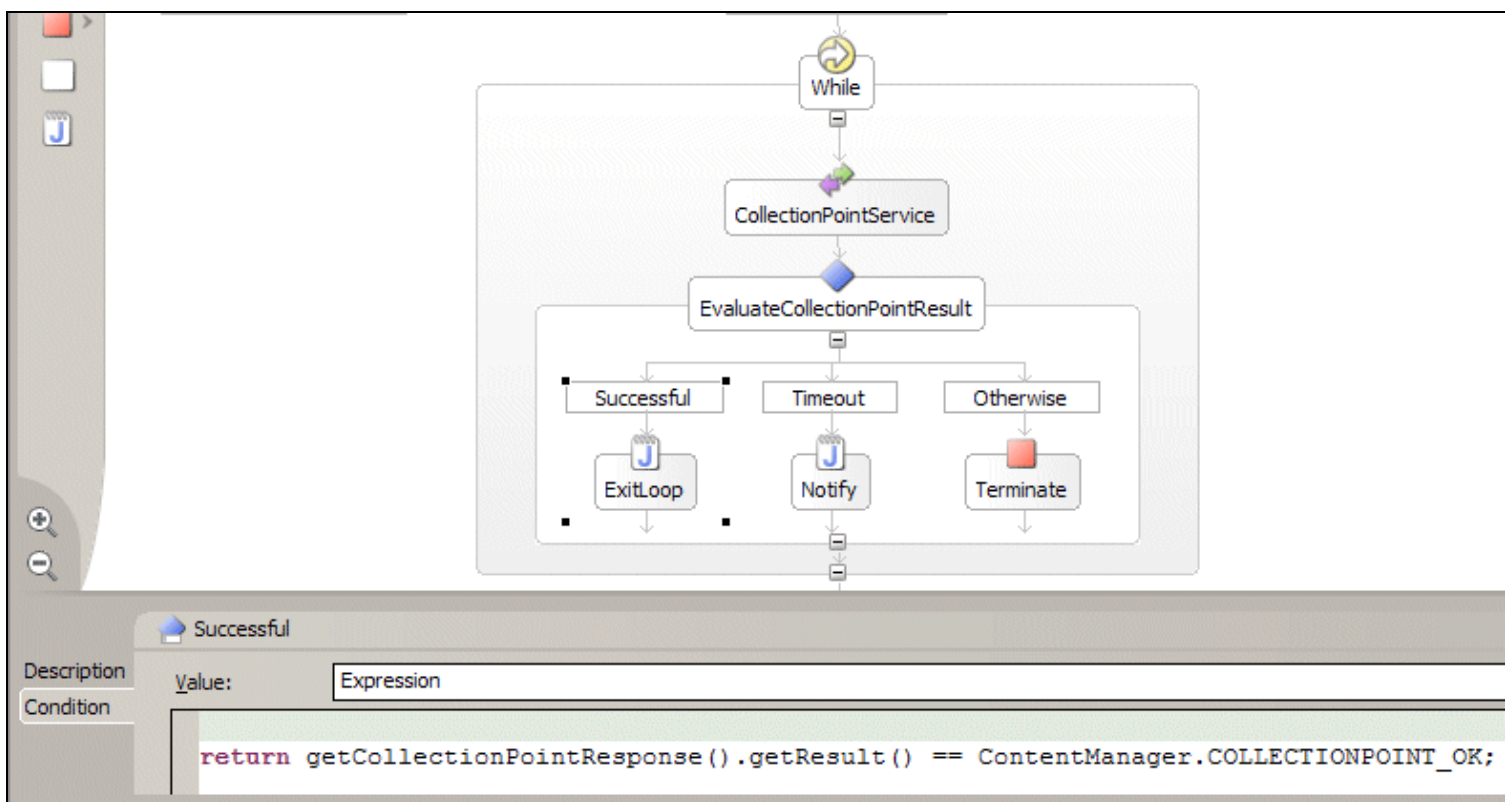
The collection point service takes an input message consisting of two arrays with the item type names and their respective quantities and a numeric part for the timeout. It returns a numeric status value that represents one of the following situations:

- *Success* - in this case the condition is met by the folder content
- *Timeout* - the condition has not been met within the number of milliseconds specified by the timeout parameter of the input message
- *Folder has deleted* - the folder that is routed through the process (on which the collection point condition is defined) has been deleted
- *Failure* - an exception has occurred while the process instance was waiting for the collection point condition to be met

A typical implementation of the collection point looks as shown below:



The Java snippet `InitializeCollectionPoint` instantiates the input message of the collection point service. It stores the item PID of the folder, and defines the collection point condition and timeout value as shown above. A decision inside the loop evaluates the return value of the collection point service and triggers corresponding actions such as sending a notification in case of a timeout or terminating the process instance in case of a failure.



A collection point service can be added to a process definition by dragging the `CollectionPoint.wsdl` from `ContentManagerIntegrationJar/com.ibm.bpe.cm/` and dropping it into the appropriate location of the process diagram. Since the collection point condition is specified by the input message of the collection point service, it can be read and modified at process runtime. The `ValidateAndAdjust` activity of the sample process demonstrates how a person with this role can adjust the collection point conditions within a certain range.

How the Collection Point Service is Implemented

Invoking a collection point service produces a JMS message that triggers the registration of a collection point. For each message, `registerCollectionPoint()` retrieves the parameters initialized by the process and writes them to a table `BPECOLLECTIONPOINTITEMTYPES` in the Content Manager library server using JDBC.

The Content Manager library server database contains two tables for collection point support provided by the Integration Toolkit.

The table `BPECOLLECTIONPOINTITEMTYPES` contains one row for each item type and quantity specified in the collection point request. It is defined as follows:

| | |
|------------------------------|----------------------------|
| <code>CORRELATIONID</code> | <code>VARCHAR (254)</code> |
| <code>ITEMYPENAME</code> | <code>VARCHAR (254)</code> |
| <code>ITEMYPEQUANTITY</code> | <code>INTEGER</code> |

The `CORRELATIONID` column contains the JMS Correlation ID of the original message. The `ITEMYPENAME` column contains the name of the item type to wait for. `ITEMYPEQUANTITY` specifies the number of items of type `ITEMYPENAME` to wait for.

The table `BPECOLLECTIONPOINTS` contains one row for each active collection point request. It is defined as follows:

| | |
|----------------------------|----------------------------|
| <code>CREATETS</code> | <code>TIMESTAMP</code> |
| <code>CORRELATIONID</code> | <code>VARCHAR (254)</code> |
| <code>ITEMID</code> | <code>VARCHAR (254)</code> |

The `CREATETS` column is the creation date and time of the collection point request. It ensures that collection point requests are processed in the order in which they are created. The `CORRELATIONID` column is the JMS Correlation ID of the original message. Process Choreographer uses it to correlate the response message with the request message so that the Invoke activity in the process completes. The `ITEMID` column is the Item ID of the folder to be checked.

There is a one-to-many relationship between `BPECOLLECTIONPOINTS` and `BPECOLLECTIONPOINTITEMTYPES`, based on the `CORRELATIONID`. Any of these tables is created during server startup if it does not already exist.

The method `checkCollectionPoint()` uses JDBC to perform an SQL query which returns a list of collection points that are satisfied. For each satisfied collection point, it deletes the collection point request from the `BPECOLLECTIONPOINTS` table and sends a response message back to the Collection Point Invoke activity of the process. A request is also removed if the folder that is routed through the process has been deleted before the collection point condition is satisfied. Each row in the `BPECOLLECTIONPOINTS` table represents a collection point, and each row in the `BPECOLLECTIONPOINTITEMTYPES` table represents an item type condition for a collection point. `checkCollectionPoint()` joins these tables with the Content Manager Links, Items and NLS Keywords tables to efficiently identify all satisfied collection points in a single request to the library server database. `checkCollectionPoint()` uses the "Folder Contains" link type to determine the contents of folders. It sums the items for the specified Content Manager folder, and if the resulting counts are greater than or equal to the quantities specified in the collection point conditions, then the collection point is considered to be satisfied. The NLS Keywords table maps the Content Manager item type names to numeric Item Type ID's using the base name for the item type, which may be expressed in any language. However, to avoid a potential cross product, the join is limited to `ENU`, which must be present (though not necessarily in use). The collection point tables are created (if they don't already exist) via the `ejbCreate()` method, the first time the Content Manager Integration Stateless Session Bean is created. Rows are written to the `BPECOLLECTIONPOINTITEMTYPES` by `registerCollectionPoint()`. `checkCollectionPoint()` is invoked from the Content Manager Integration Asynchronous Bean.

4.5 Content Attribute Access

There are a number of situations in which there is a need to access the attribute values of the item that is routed through the process or of items in the folder that is routed through the process. Typical situations are conditions on links in a parallel section (flow) of the process or on switch statements in a 'case' section that determine the path to be taken depending on the value of an attribute. Another typical situation is instantiating the input message of a staff activity or other service by extracting attribute values from the folder or from an item in the folder and placing them into parts of the input message to make them available as input parameters to the service. The integration toolkit offers the following methods to extract attribute values from items or to evaluate conditions over attributes:

```
Object ContentManager.getContentAttribute(processBean, attributeSpec)
Object [] ContentManager.getContentAttributes(processBean, attributeSpec)
Object ContentManager.getFolderContentAttribute(processBean, attributeSpec)
```

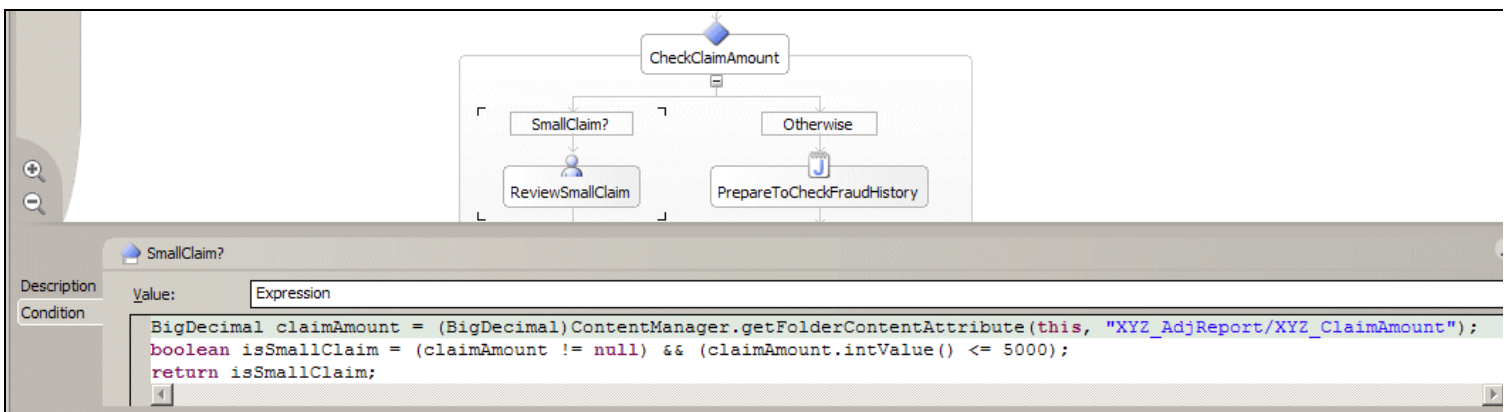
```
Object [] ContentManager.getFolderContentAttributes(processBean, attributeSpec)
boolean ContentManager.evaluateCondition(String XPath)
boolean ContentManager.evaluateCondition(ProcessBean bean, String XPath)
```

Each of these methods takes a reference to the process bean and an attribute specification as input parameters. The attribute specification is an array of Strings of the form <itemTypeName>/<attributePath>, where <itemTypeName> specifies the type of item that holds the attribute value of interest (the same attribute may be defined for many different items). <attributePath> can be an atomic attribute name or a sequence of attribute names separated by '/' in case of an attribute hierarchy (as for example in customer/address/city).

getContentAttribute() and getFolderContentAttribute() can be used if no more than a single value can be returned from the query (if the queried attribute is single-valued and no more than a single item of this type is contained in the folder). getContentAttributes() and getFolderContentAttributes() return arrays of objects and are thus capable of returning multiple values.

Decision points are an example for using attribute value access. Many business processes contain conditional processing, in which the path through the process is controlled by the value of attributes or variables in the process. Furthermore, the links between activities in a parallel section (flow) can be conditional, meaning that the link is only followed if the condition is true. These conditions can be specified in a variety of ways, including Java expressions. To use a decision point, a process designer adds a link between activities, specifies the value of the condition as Expression, and enters a Java expression that uses one of the methods listed above to evaluate the condition.

Here is an example of a Decision Point condition based on getFolderContentAttribute(). The condition verifies if the value of the attribute XYZ_ClaimAmount exceeds a certain limit. The query looks for this value in items of type XYZ_AdjReport located in the folder which is routed through the process. If the attribute would be multi-valued or more than one item of this type could be part of the folder, getFolderContentAttributes() should be used instead because getFolderContentAttribute() accesses only the first element of the result set.

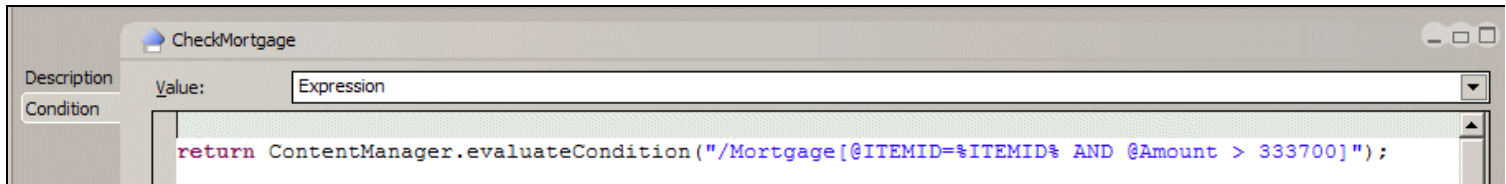


evaluateCondition() is based on a different concept to implement conditions on links or switch statements. When called with the process bean and an XPath expression as parameters, it returns true if the result of the query is non empty. The evaluateCondition(ProcessBean bean, String XPath) form of this method supports the following substitutable parameters, which can be included in XPath expressions to further qualify the results:

```
%ITEMID%
%COMPONENTID%
%VERSIONID%
```

These substitutable parameters are replaced with the values of the Item ID, Component ID and Version ID (respectively) of the CM item associated with the process (typically a folder). The Item ID and Component ID parameters are character strings, and the substituted values include quotes to produce a valid XPath expression. The Version ID parameter is a number, which can be used in numeric expressions (e.g. %VERSIONID% > 3). Note that use of these substitutable parameters is optional and that the XPath expression is not limited to querying attributes of items associated with the process.

Here is an example of a Process Choreographer conditional link expression that checks to see if the mortgage amount exceeds \$333,700:



This Java expression uses the `%ITEMID%` substitutable parameter to get the CM Item ID associated with the process. This Item ID qualifies the search so that it only applies to the CM item associated with the current process. In this example, the item type is Mortgage and the name of the CM attribute is Amount. If the mortgage amount exceeds \$333,700 for the item associated with the current process instance, then the link will be followed and the next activity on that link will be executed.

4.6 Folder Service

The folder service adds one or more items to a folder. The two available operations are `addFlowItemToFolder` and `addItemToFlowFolder`. Both operations take the `itemId` of the item that is routed through the folder and an XPath query as input and return a numeric value that represents the size of the result list obtained from running the XPath query. The operations differ as follows:

`addFlowItemToFolder`

In this case the XPath query needs to return a folder. This operation adds the item that is routed through the process to this folder. If the item routed through the process is itself a folder this will create a nested folder structure.

`addItemToFlowFolder`

In this case the item routed through the process needs to be a folder. This operation adds the items returned by the XPath query to the folder that is routed through the process.

4.7 Combined Query

Typically, end users are not aware of underlying technologies and need not be concerned with whether they are using a process management system or a content repository. Ideally, these users have a single user interface that enables them to efficiently search for information, regardless of how that information is stored. In order for this to be the case, the Integration Toolkit provides a combined query capability that efficiently searches for information in both the Content Manager library server and the process engine database (i.e. not only content and its attributes, but also properties of the enclosing process instance). This query capability utilizes search criteria for both systems (the "where" clause) as well as attributes (the "select" clause). Combined query results only include items to which the caller has access. This means that the item must exist in a work list, and that the caller be authenticated as a user who has access to the work item.

To take advantage of this combined query capability, a Web programmer uses the `combinedQuery` method of the Integration Toolkit which has the following interface:

```
public QueryResultSet combinedQuery(
    String selectClauseWorkflow,
    String flowItemTypeNameContent,
    String flowItemAttributesContent,
    String additionalAttributesContent,
    String whereClauseWorkflow,
    String xpathContent,
    String orderByClauseWorkflow,
    java.util.TimeZone timezoneWorkflow);
```

The `combinedQuery` method returns a `com.ibm.bpe.api.CombinedQueryResultSet` which enables a programmer to access the combined query results. This interface is designed to be used e.g. in a Web application. The Integration Sample Client implements the concept of user work lists based on the combined query capability of the Integration Toolkit. below.

4.8 Staff Resolution

As work flows through a process management system, it is acted on by a variety of automated and manual steps. When a work item arrives at a manual step, the process management system must decide which users should have access to the item. This is referred to as staff resolution. Process Choreographer provides a highly configurable staff resolution facility that uses a role / verb model to identify user access for a work item. These are the predefined roles that describe the types of access supported:

- *Administrators* - People response for upper level administrative tasks

- *Potential Owners* - People who may claim the item, perform the activity, and complete it
- *Editors* - People who may update the item, but not complete the activity
- *Readers* - People who may only view the activity, but not work on it

Process Choreographer uses staff verbs to assign users to these staff roles. The staff verbs are configurable queries to an underlying directory service, and are limited only by the power and flexibility of that service. For example, if the directory service supports a "manager of" relationship, then the staff verb can utilize that relationship to resolve a role to a manager of a user. Process Choreographer utilizes a pluggable interface to the underlying staff repository, and comes preconfigured with interfaces for LDAP, the WebSphere User Repository and a System directory.

The Integration Toolkit includes a WebSphere User Registry implementation for DB2 Content Manager. This uses the Content Manager user and group definitions to support group membership. This means that users only need to be defined once, using the Content Manager system administration interface, and that staff resolution can be performed at the group level. Alternatively an LDAP server can be used as a centralized repository for user and group definitions as will be done in [Chapter 5](#) of this document.

Note that the WebSphere user registry for DB2 Content Manager provided by the Integration Toolkit can not be used to implement Single Sign On (SSO) since enabling Content Manager for SSO requires disabling of the password checks. When used with an SSO-enabled application server this would bypass password checking and allow users to log on with an arbitrary password. The preferred way to implement SSO is by using a simple file-based user registry in the build environment (based on the files users.prop and groups.prop) and to use an LDAP system such as the Tivoli Directory Server for the runtime environment.

Note: Using a custom WebSphere User Registry affects all application running on the Application Server such as the Administrative Console. With a dedicated WAS installation for Content Manager and Process Choreographer (potentially also hosting the Resource Manager application) this should typically not be a problem as we recommend to use the library server userid (icmadmin) as Application Server and process administrator. If the Application Server is intended to host other applications as well, potential side effects when using a custom user registry need to be well-understood.

4.9 Customizing the Quick Start Client

The Quick Start Client is a generic DB2 Content Manager and Process Choreographer client that does not cover the full range of capabilities offered by DB2 Content Manager and Process Choreographer but it provides what can be thought of as a reasonable set of the most typically used functions and since it is available as source code it can be adjusted to the specific needs of your application.

The process-specific information is stored in QuickStartClient\JavaSource\com\ibm\bpe\cm\util\ProcessData.java. The following settings may need to be adjusted:

Global constants

- CM_DATASTORE_NAME ... the name of the library server database (e.g. icmnsdb)
- AUTO_LINK_ATTRIBUTE ... the name of the attribute that defines folder membership if a folder is routed through the process

QuickStartClient\JavaSource\com\ibm\bpe\cm\util\Helpers.java contains a method `prepareName()` that can be used to map a display name to some potentially truncated form that displays well on the client pages. For the First Steps sample data this means cutting off the comment in parenthesis: (Content Manager V8.1 Sample Item Type) or to truncate the display name of the XYZ_ClaimFolder Claim Application Folder to contain other claim related documents (e.g. adjuster report, auto photos, claim form, etc) and linked by Claim Number at character position 24. This method may be helpful in cases like this where the display name serves as a 'description' of the item type in addition to being the (localizable) name that is displayed to users. `prepareName()` assumes display names have the form <actual display name> (<description>) where <actual display name> is the name to be shown in the client and (<description>) is the comment that describes the item type in more detail. The constants `DISPLAY_NAME_SIZE_LIMIT` and `DISPLAY_NAME_TRUNCATION_POS` can be used additionally to truncate long names (those the length of which exceeds `DISPLAY_NAME_SIZE_LIMIT`) to a maximum size of `DISPLAY_NAME_TRUNCATION_POS`. Note that specific rules may need to be applied depending on the locale (see the section on [Globalization and Accessibility Considerations](#) for more details on this topic).

Roles

Roles define certain aspects of the client's look and feel. A user who logs on to the client gets a role ID assigned by one of the following two mechanisms:

- a. it is passed as a parameter to `listWorkItems.jsp` with a URL such as `http://localhost:11605/QuickStartClient/listWorkItems.jsp/role=5`. This is the preferred option used with the `ClaimsHandlingProcess` sample.
- b. a map translates user IDs into role IDs (see the comments in `header.jsp` for an example of how this could be done)

Role IDs are used as an index into three arrays: `workListStyleMap`, `addToFolderActionItemTypes`, and `addToFolderActionNames`. They should be defined using symbolic constants as in the following example:

```
static public final int USER1_ROLE = 0;
static public final int USER2_ROLE = 1;
static public final int NUM_OF_ROLES = 2;
static public final int DEFAULT_ROLE = USER1_ROLE;
```

The `workListStyleMap` defines the columns of a worklist for this role in terms of process properties and item attributes. A `WorkListStyleMap` defines the parameter of the combined query to be run when retrieving work items for this role.

`addToFolderActionItemTypes` and `addToFolderActionNames` define which documents a user with this role can add to the folder that is routed through the flow. For example

```
addToFolderActionItemTypes [USER1_ROLE] = { "AdjReport", "FraudReport",
"ReliabilityStudy" };
addToFolderActionNames [USER1_ROLE] = { "Adjuster Report", "Fraud Report",
"Reliability Study" };
```

...means that a user with role `USER1_ROLE` may add instade of the item types `AdjReport`, `FraudReport`, `ReliabilityStudy` to the folder that is routed through the flow. The action column of the work page will refer to these actions by 'Add Adjuster Report', 'Add Fraud Report', 'Add Reliability Study' based on the elements of `addToFolderActionNames`.

User Actions

The method `staffActions(String activityName, WSIFDefaultMessage inputMsg)` defines the display elements for the 'Process activity' section of the work page. This section is defined as a table segment spanning four columns. There is no limit on the number of rows. The final column always contains the available actions (Complete and Cancel). This section typically displays some parts from the input message of this staff activity (2nd parameter) and contains form elements that can be used to instantiate its output message. The name of the form element needs be of the form `msgpart.xxx` where `xxx` is the name of the corresponding message part. If all parts of the output message are of type `String`, no custom logic needs to be added to the servlet `CompleteWorkServlet` that maps the form values to the output message. If non-string-typed variables are involved, either output message parts are mapped to the target type by a Java snippet in the process or custom logic needs to be added to the `CompleteWorkServlet`

Here a simple example of a 'review activity' with an input message that contains the text to be reviewed and an output message consting of a rating in the range 1..5.

```
static public String staffActions(String activityName, WSIFDefaultMessage inMsg) {
    try {
        String result = "<td colspan=\"4\"></td>"; // default

        if (activityName.equals("Review")) {
            String textToBeReviewed = (String)inMsg.getObjectPart("TextToBeReviewed");
            result = "<TD colspan=\"2\">\n" +
                "Text to be reviewed:" + textToBeReviewed +
                "</TD> " +
                "<TD colspan=\"2\">\n" +
                "<SELECT name=\"msgpart.NumAdjReports\"> " +
                "<OPTION value=\"1\" SELECTED>1</OPTION>" +
                "<OPTION value=\"2\">2</OPTION>" +
                "<OPTION value=\"3\">3</OPTION>" +
                "<OPTION value=\"4\">4</OPTION>" +
                "<OPTION value=\"5\">5</OPTION>" +
            "
```

```

                                "</SELECT>" +
                                "</TD>";
        } // 'Review' activity
        else if (activityName.equals("SomeOtherActivity")) {
            ....
        }
    } catch (WSIFException e) {
        e.printStackTrace();
    }
    return result;
}

```

4.10 The project structure of a custom integration solution

To create a custom integration solution the following projects need to be imported from `quickStartSampleRev1wbi.zip`:

- ContentManagerIntegration
- ContentManagerIntegrationEjb
- ContentManagerIntegrationJar
- QuickStartClient
- Server

Make sure that the properties of the service project that hosts the custom process includes `ContentManagerIntegrationJar` in the list of project references and in the list of projects that participate in the build path. `CollectionPoint` or `FolderManagement` services can be added to the process definition by dragging their `.wsdl` file from `ContentManagerIntegrationJar\com\ibm\bpe\cm` and dropping at the appropriate position of the process flow and adding the corresponding Java snippets or assign activities that create and interpret in- and output messages of the service. When generating deploy code for the process the corresponding `XXXService.wsdl` files need to be located in the 'Referenced Partners' section of the **Generate Deploy Code** window. See the section on how to [Resolve BPEL references by binding services to port types](#) for details.

4.11 Globalization and accessibility considerations

All Web pages of the Quick Start Client use UTF8 encoding. The static content can easily be localized by creating translated versions of these pages. To avoid code duplication for JSPs we recommend using a different approach for most of the JSPs except `help.jsp`, namely using Java Resource Bundles since most JSPs only contain small portions of static text such as banner or column titles. A similar approach needs to be taken to internationalize messages of the Integration Toolkit such as those thrown in case of an `IntegrationException`.

Note that using images to represent the process flow as in `index.jsp` requires a corresponding text-only version to meet accessibility requirements.

5. Deploying an Integrated Solution on a Production Simulation Environment

This chapter describes the steps needed to set up a stand-alone environment that can be used to simulate and test the process in a system configuration that matches a production environment. This is done by deploying the process, Integration Toolkit runtime and Quick Start client on a WebSphere Business Integration Server Foundation, using an LDAP server as a common user repository and storing the Process Choreographer data in a DB2 database. Depending on previous experience with LDAP and the WebSphere server environment completing this paragraph will take about 2 hours.

5.1 Prerequisites and Installation Hints

See [1.3 Software Requirements](#) for details on the system configuration required for chapter 5.

Configuring the business process container

The Quick Start Integration assumes both, the Content Manager data and Process Choreographer data reside in DB2. Therefore the business process container has to be configured to use the DB2 Universal Database.

- Make sure DB2 is installed and `server 1` is running.
- Optionally: investigate the process choreographer documentation about information on table spaces for the Process Choreographer database. Creating a custom tablespace for Process Choreographer may be an advantage from a performance optimization perspective.

- In <WAS_HOME> run `bin\wsadmin -f ProcessChoreographer\sample\bpeconfig.jacl`. If you want to unconfigure an existing configuration, you may need to use `bpeunconfig.jacl` first. If security is active you need to add the parameters `-user <ICMADMIN> -password <ICMADMINPW>`.

```

C:\> Command Prompt - wsadmin -f D:\WebSphere\AppServer\ProcessChoreographer\sample\bpeco...
r\sample\bpeconfig.jacl
WASX72091: Connected to process "server1" on node triangle using SOAP connect
The type of process is: UnManagedProcess

*****
* This script allows to install bpecontainer.ear, install the Web client,
* create the Process Choreographer DB, create the WebSphere MQ queues, and
* update WCCM with the DataSource, ListenerPorts, and Scheduler required by
* Process Choreographer.
* Supported databases are Cloudscape, DB2, Informix, Sybase, Oracle, and
* SQL Server; supported JMS providers are WebSphere Embedded Messaging and
* WebSphere MQ.
* The prerequisite software must already be installed.
*
* You will be prompted for the required information at each step. The
* default value is always listed first in a prompt, you can select it by
* simply pressing the 'Enter' key.
*
* DISCLAIMER: This sample script is provided AS-IS for your convenience. It
* can only create simple configurations. If you need to create more complex
* configurations, please follow the instructions provided in the InfoCenter.
*****

More than one server found. Please specify where to configure Process Choreog
her.
Configure on server 'cmwebsvc' of node 'triangle' [Yes/no?

```

- Make sure Process Choreographer is configured on `server1` since more than one server may be present (e.g. `icmrm`).
- Defaults are appropriate except for the following settings:

| Name | Value |
|---|---------------------------------------|
| Server where Process Choreographer is configured | server1 |
| BPESystemAdministrator | icmadmin |
| JMSAPIUser | icmadmin |
| database | DB2 |
| Server of Process Choreographer to connect to | server1 |
| DB2 user ID | depending on your system |
| JMSAPIUser | icmadmin |
| WebSphere MQ user ID | icmadmin |
| Enable global security using the Local OS user registry | no (this will be done manually later) |
| Enforce Java 2 security | no |

- Optionally: investigate `<WAS_HOME>\logs\bpeconfig.log` to ensure the settings are correct.
- (Re)start `server1`.

5.2 Setting up LDAP

Refer to the product documentation ([DB2 Content Manager V8.3 Information Center](#)) and follow the instructions *Planning and Installing Your Content Management System > After-installation configuration and setup procedures > Integrating with business processes > Part2: Setting up a production environment > Preparing a common organizational directory* to perform the following steps:

- Import the directory data into *Tivoli Directory Server*.
- Configure DB2 Content Manager for LDAP. For this topic we recommend the section *Planning and Installing Your Content Management System > Using DB2 Content Manager after-installation programs and procedures > Enabling LDAP*.

- Configure WebSphere Application Server for LDAP.
- Configure the LDAP staff plug-in for Process Choreographer.
- Follow the steps in *Planning and Installing Your Content Management System > After-installation configuration and setup procedures > Part2: Setting up a production environment > Configure the business process for use in the production environment > Enable the business process for LDAP*. Note that the name of the business process is now ClaimsHandlingProcess instead of WPCAutoClaimsProcessing.

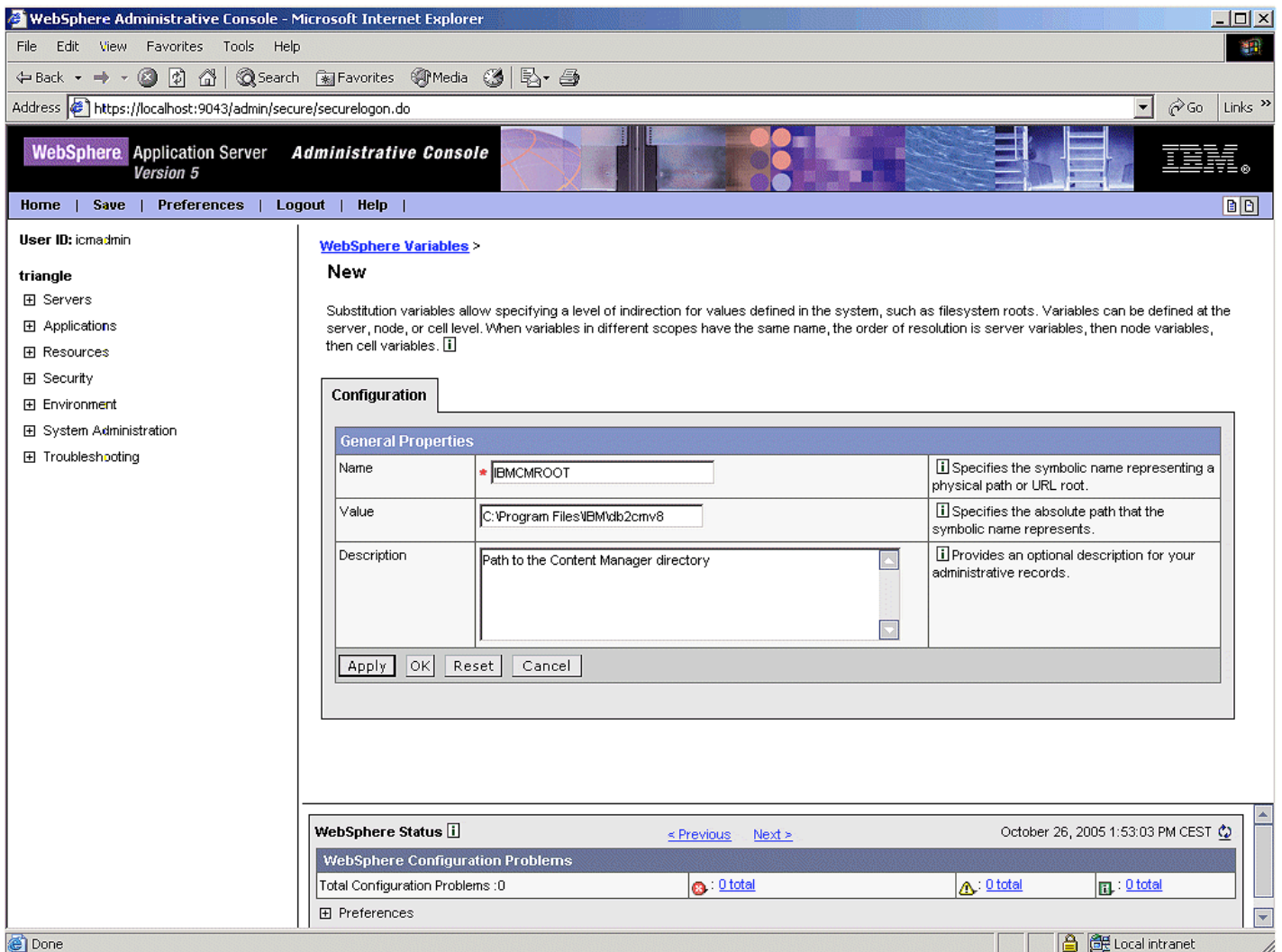
5.3 Configure the server environment

- Launch the WebSphere Administrative console by loading `<http://<HOSTNAME>:9090/admin` in your browser and log on as administrator.
- In the navigation panel select Environment > Manage WebSphere Variables and set the scope to server1.
- Ensure that the variable DB2_JDBC_DRIVER_PATH is present and that it points to `<DB2HOME>/java`

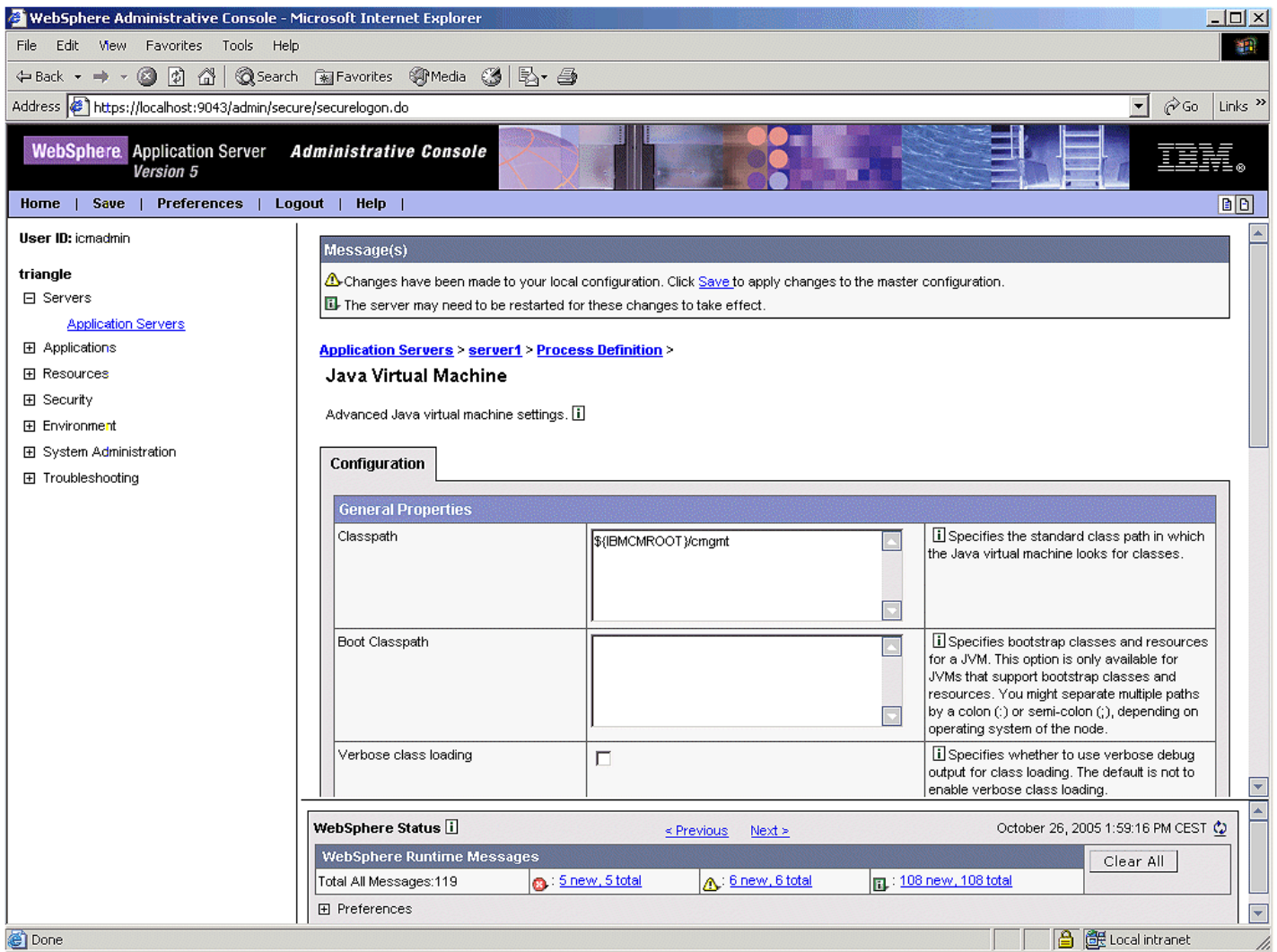
The screenshot shows the WebSphere Administrative Console interface. The left navigation pane is expanded to 'Environment' > 'Manage WebSphere Variables'. The main content area shows the configuration for the 'triangle' cell and node. A table lists several variables, with 'DB2_JDBC_DRIVER_PATH' selected. The table has columns for Name, Value, and Scope.

| Name | Value | Scope |
|-------------------------------|---|-------------------------------|
| APP_INSTALL_ROOT | \${USER_INSTALL_ROOT}\installedApps | cells:triangle:nodes:triangle |
| CLOUDSCAPE_JDBC_DRIVER_PATH | \${WAS_INSTALL_ROOT}\cloudscape\lib | cells:triangle:nodes:triangle |
| CONNECT.JDBC_JDBC_DRIVER_PATH | | cells:triangle:nodes:triangle |
| CONNECTOR_INSTALL_ROOT | \${USER_INSTALL_ROOT}\installedConnectors | cells:triangle:nodes:triangle |
| DB2390_JDBC_DRIVER_PATH | | cells:triangle:nodes:triangle |
| DB2UNIVERSAL_JDBC_DRIVER_PATH | | cells:triangle:nodes:triangle |
| DB2_JDBC_DRIVER_PATH | d:\sqlib\java | cells:triangle:nodes:triangle |

- Click **New**, enter IBMCMROOT into the **Name** field, `<IBMCMROOT>` into the **Value** field, and click **OK**

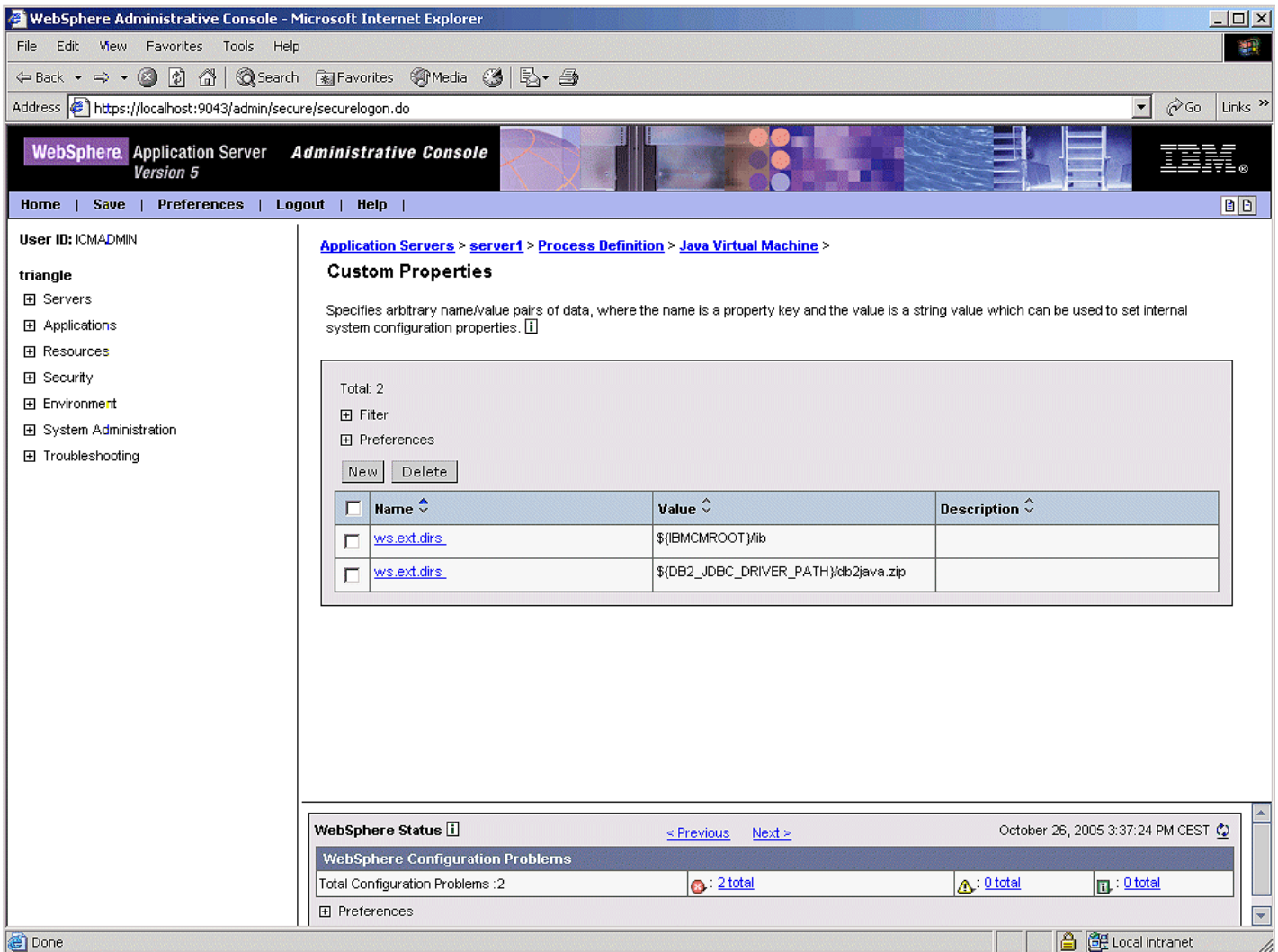


- In the navigation panel select Servers > Application Servers, click **server1** and follow the Additional Properties to **Process Definition > Java Virtual Machine**.
- Enter `${IBMCMROOT}/cmgmt` into the **Classpath** field and click **Apply**



- In the Additional Properties section select **Custom Properties**.
- In the **Custom Properties** section add the following entries by clicking **New** entering the data and clicking **OK**:

```
ws.ext.dirs          ${IBMCMROOT}/lib
ws.ext.dirs          ${DB2_JDBC_DRIVER_PATH}/db2java.zip
```

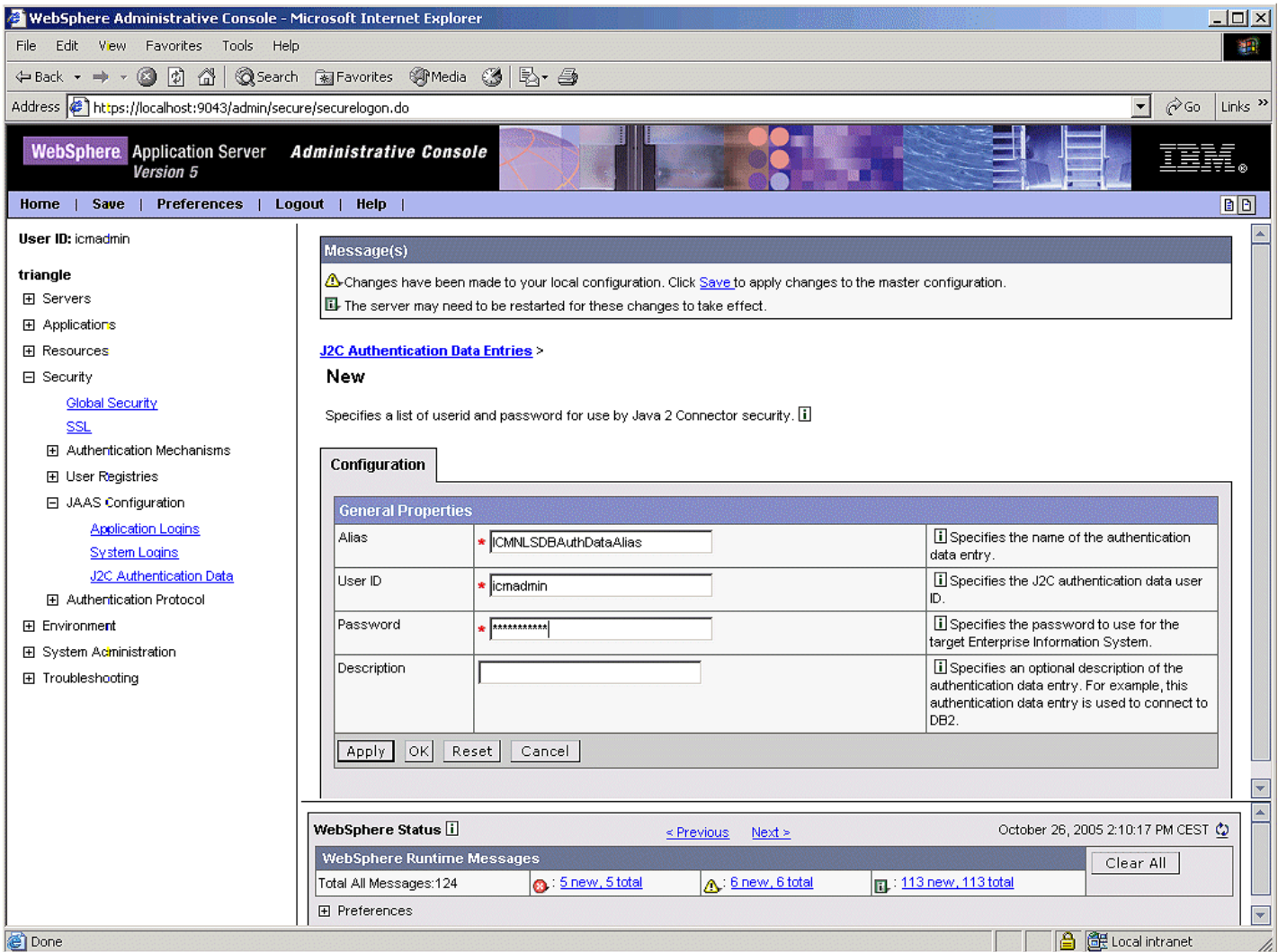


- Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.

5.4 Configure JDBC connections and data sources

- In the navigation panel select **Security** > **JAAS Configuration** > **J2C Authentication Data**.
- Verify the existence of the authentication alias `<HOSTNAME>/BPEAuthDataAliasEmb_<HOSTNAME>_server1` that has been created during process container configuration.
- Click **New**, add the following entries and click **OK**

| | |
|-----------------|-----------------------|
| Alias | ICMNLSDBAuthDataAlias |
| User ID | icmadmin |
| Password | icmadmin's password |



- In the navigation panel select Resources > JDBC Providers. Ensure the scope is on server1.
- Verify the existence of the JDBC driver DB2 Legacy CLI-based Type 2 JDBC Driver (XA) that has been created during process container configuration. Click on it.
- In the Additional Properties section of the configuration panel select **Data Sources**, Verify the existence of the data source BPData sourceDb2 that has been created during process container configuration.
- Select **New** to create a new data source based on the following information:

| | |
|---|--|
| Name | replace DB2 Legacy CLI-based Type 2 JDBC Driver XA Data Source with icmnlbdb |
| JNDI Name | jdbc/ICMNLSDDB |
| Component-managed Authentication Alias | select <HOSTNAME>/ICMNLSDDBAuthDataAlias from the list |
| Container-managed Authentication Alias | select <HOSTNAME>/ICMNLSDDBAuthDataAlias from the list |
| Description | CM library server data source for CM/PC Integration Quick Start |
- Click **OK** to save the data source definition.

WebSphere Administrative Console - Microsoft Internet Explorer

Address: <https://localhost:9043/admin/secure/securelogin.do>

WebSphere Application Server Administrative Console Version 5

Home | Save | Preferences | Logout | Help

User ID: icmadmin

triangle

- Servers
- Applications
- Resources
 - JDBC Providers
 - Generic JMS Providers
 - WebSphere JMS Provider
 - WebSphere MQ JMS Provider
 - Mail Providers
 - Resource Environment Providers
 - URL Providers
 - Resource Adapters
 - Cache Instances
 - Extended Messaging Provider
 - Object Pools
 - Scheduler Configuration
 - Staff Plugin Provider
 - Work Manager
 - WebSphere Business Integration Adapter
- Security
- Environment
- System Administration
- Troubleshooting

Message(s)

Changes have been made to your local configuration. Click [Save](#) to apply changes to the master configuration.
 The server may need to be restarted for these changes to take effect.

[JDBC Providers](#) > [DB2 Legacy CLI-based Type 2 JDBC Driver \(XA\)](#) >

Data Sources

Data Source is used by the application to access the data from the database. A data source is created under a JDBC provider which provides the specific JDBC driver implementation class. [i](#)

Total: 2

Filter

Preferences

New Delete Test Connection

| <input type="checkbox"/> | Name | JNDI Name | Description | Category |
|--------------------------|----------------------------------|---------------|--|----------|
| <input type="checkbox"/> | BPEDataSourceDb2 | jdbc/BPEDB | DataSource for Process Choreographer | |
| <input type="checkbox"/> | icmnlsdb | jdbc/ACMNLSDb | CM library server data source for CM/BPC Integration Quick Start | |

WebSphere Status [i](#) < Previous Next > October 26, 2005 2:17:18 PM CEST [i](#)

WebSphere Runtime Messages

Total All Messages: 129 [5 new, 5 total](#) [6 new, 6 total](#) [118 new, 118 total](#) [Clear All](#)

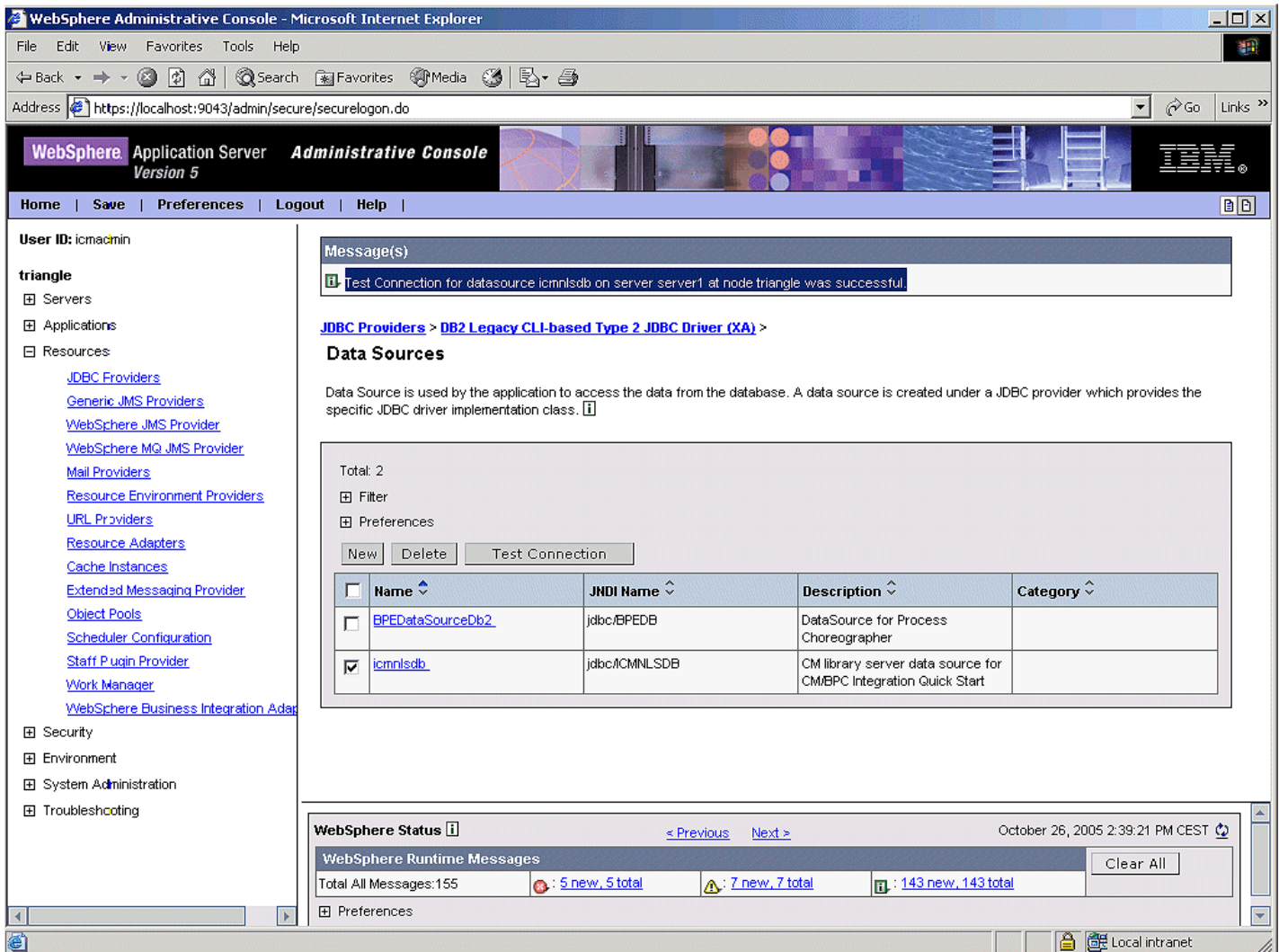
Preferences

- Click on the new **icmnlsdb** data source definition and in the Additional Properties section select **Custom Properties** and click **sample** in the value column of the Custom Properties page.

The screenshot shows the WebSphere Administrative Console interface. The breadcrumb navigation is: **JDBC Providers > DB2 Legacy CLI-based Type 2 JDBC Driver (XA) > Data Sources > Custom Properties**. The main content area displays a table of custom properties for a data source. The table has columns for Name, Value, Description, and Required. The 'databaseName' property is highlighted, with its value set to 'sample'. Below the table, there is a 'WebSphere Status' section and a 'WebSphere Runtime Messages' section showing message counts.

| Name | Value | Description | Required |
|--|--------------|--|----------|
| <input type="checkbox"/> databaseName | sample | This is a required property. The database name. For example, enter sample to make your Data Source point to jdbc:db2:sample. | true |
| <input type="checkbox"/> description | - | The description of this datasource. | false |
| <input type="checkbox"/> portNumber | - | The TCP/IP port number where the jdbc Provider resides. | false |
| <input type="checkbox"/> connectionAttribute | cursorhold=0 | The connection attributes. Refer to the DR2 reference for the list | false |

- Replace the default value `sample` in the **Value** field with `icmnlbdb` and click **OK**.
- Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.
- Select **Resources > JDBC Providers** on the navigation panel and ensure the scope is on `server1`.
- Click on **DB2 Legacy CLI-based Type 2 JDBC Driver (XA)**, scroll down to the **Additional Properties** section, click on **Data Sources** and perform the following steps:
 - Select the check box left of `icmnlbdb`.
 - Click **Test Connection**.
 - Verify that the information message on top of the page says `Test Connection for datasource icmnlbdb on server server1 at node <HOSTNAME> was successful.`

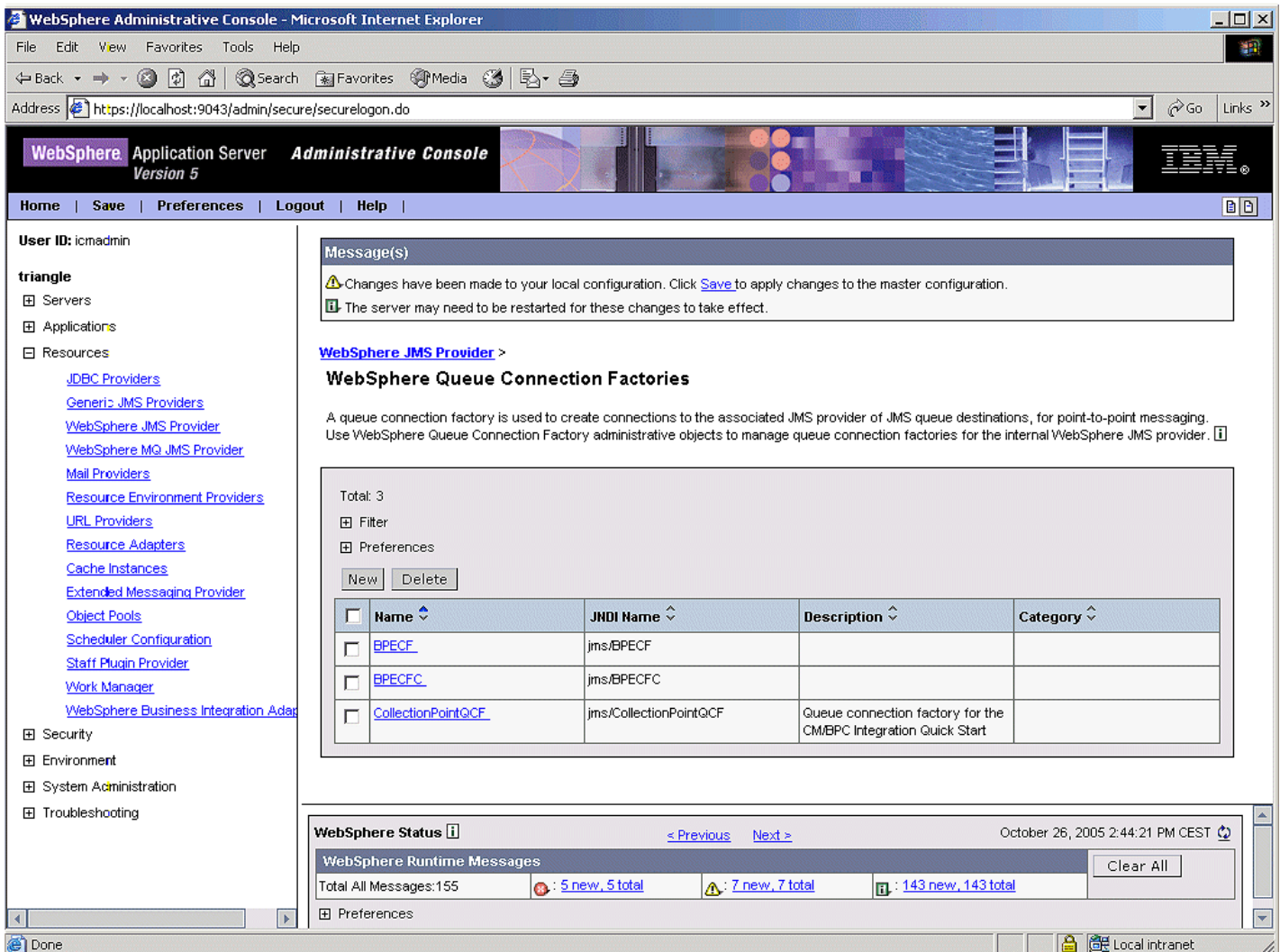


5.5 Configure connection factories, queues, and consumers

- In the navigation panel select **Resources > WebSphere JMS Provider** and set the scope to **server1**.
- In the Additional Properties section select **WebSphere Queue Connection Factories** and verify the existence of the two queue connection factories **BPECF** and **BPECF** that have been created during process container configuration.
- Click **New** to create a new queue connection factory based on the following information:

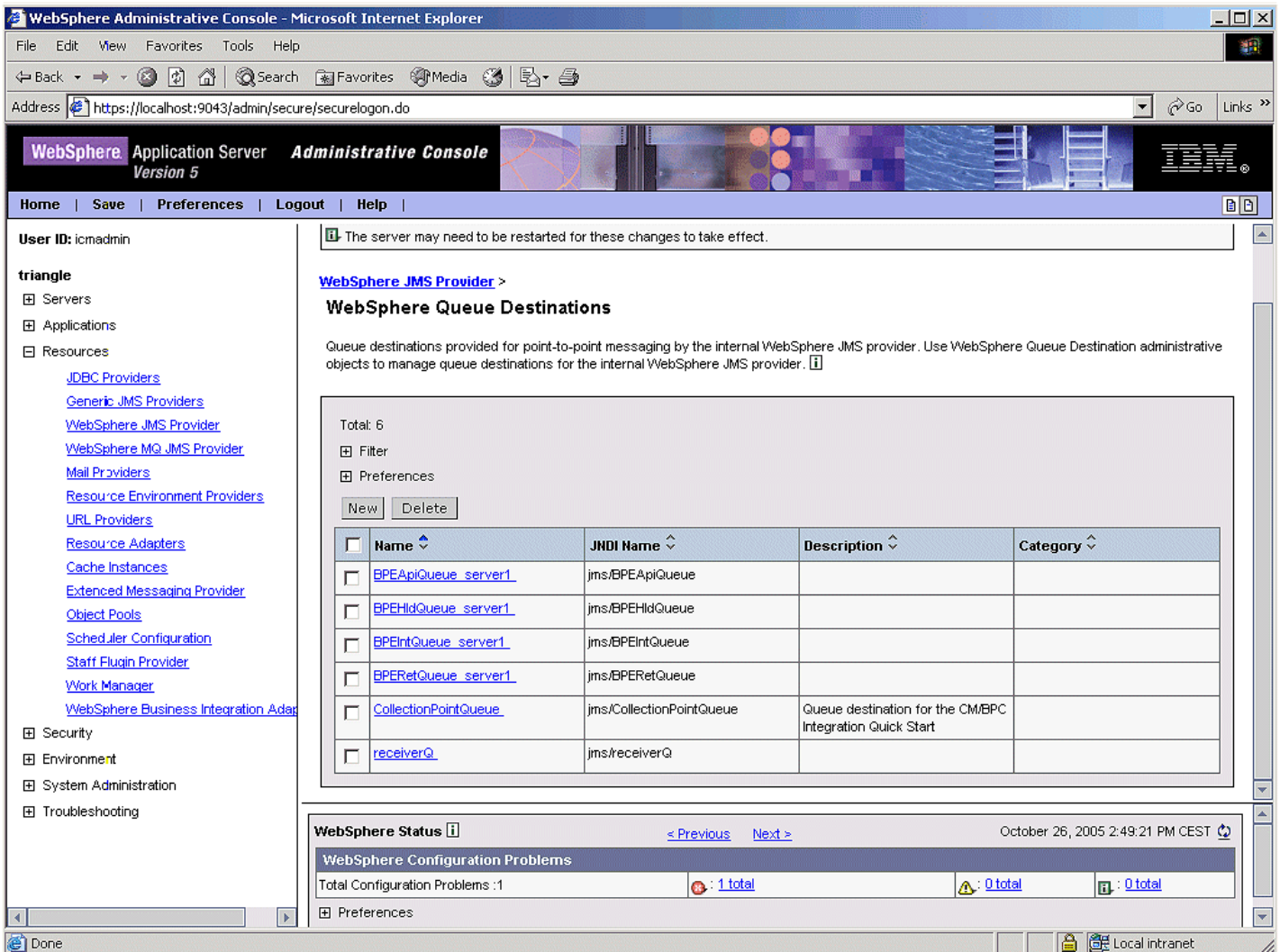
| | |
|---|--|
| Name | CollectionPointQCF |
| JNDI Name | jms/CollectionPointQCF |
| Component-managed Authentication Alias | select <HOSTNAME> / ICMNLSDbAuthDataAlias from the list |
| Container-managed Authentication Alias | select <HOSTNAME> / ICMNLSDbAuthDataAlias from the list |
| Mapping-Configuration Alias | select DefaultPrincipalMapping from the list |
| Description | optionally: Queue connection factory for the CM/PC Integration Quick Start |

- Click **OK** to save the queue connection factory definition.



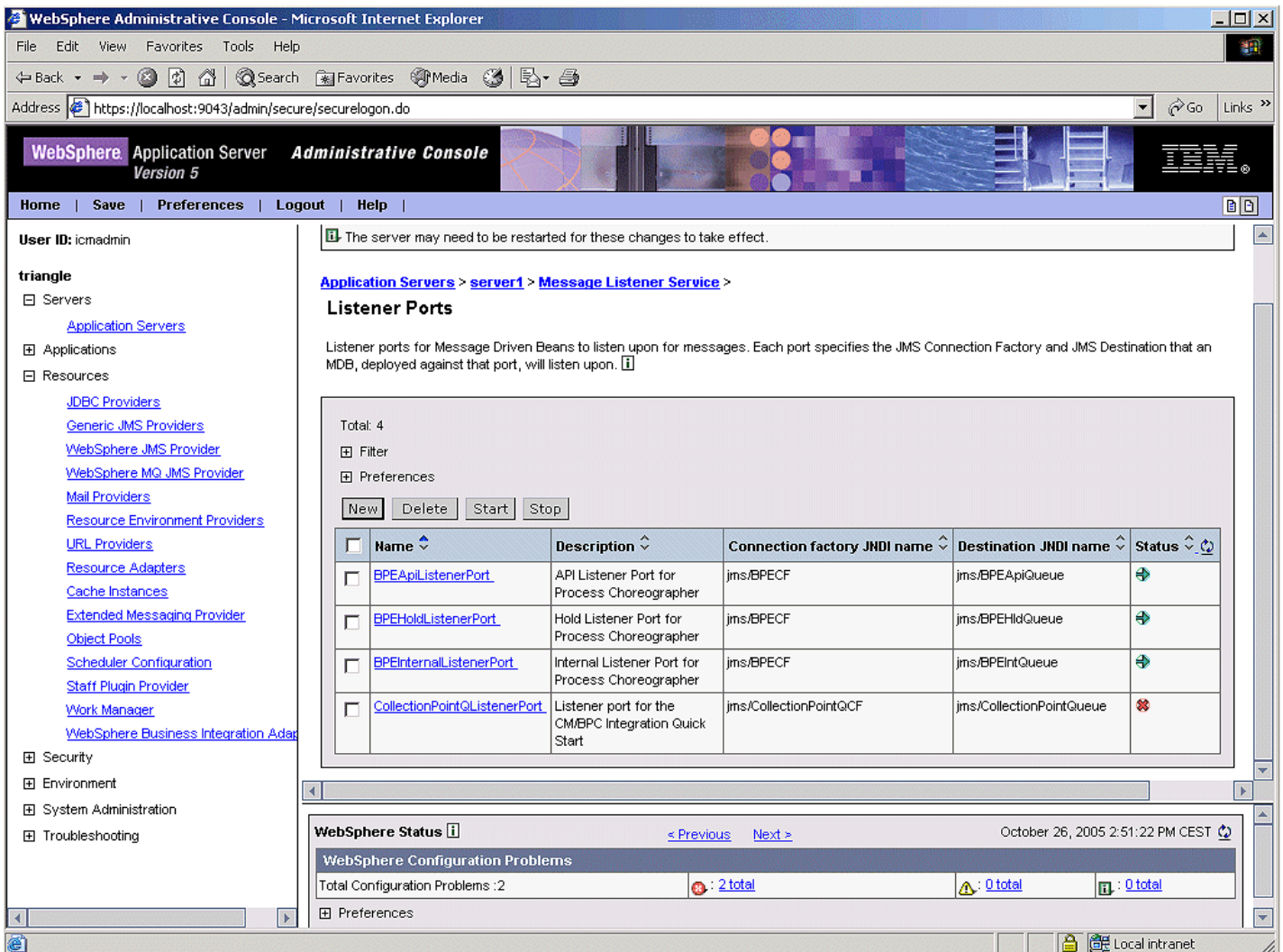
- Navigate back to the **WebSphere JMS Provider** page and click **WebSphere Queue Destinations** in the Additional Properties section.
- Verify the existence of the four queue destinations `BPEApiQueue_<SERVERNAME>`, `BPEHidQueue_<SERVERNAME>`, `BPEIntQueue_<SERVERNAME>`, and `BPERetQueue_<SERVERNAME>` that have been created during process container configuration.
- Click **New** to create a new queue destination based on the following information:

| | |
|--------------------|---|
| Name | CollectionPointQueue |
| JNDI Name | jms/CollectionPointQueue |
| Description | optionally: Queue destination for the CM/PC Integration Quick Start |
- Click **OK** to save the queue destination definition.
- Repeat the previous steps to create a new queue destination with name `receiverQ` and JNDI name `jms/receiverQ`.

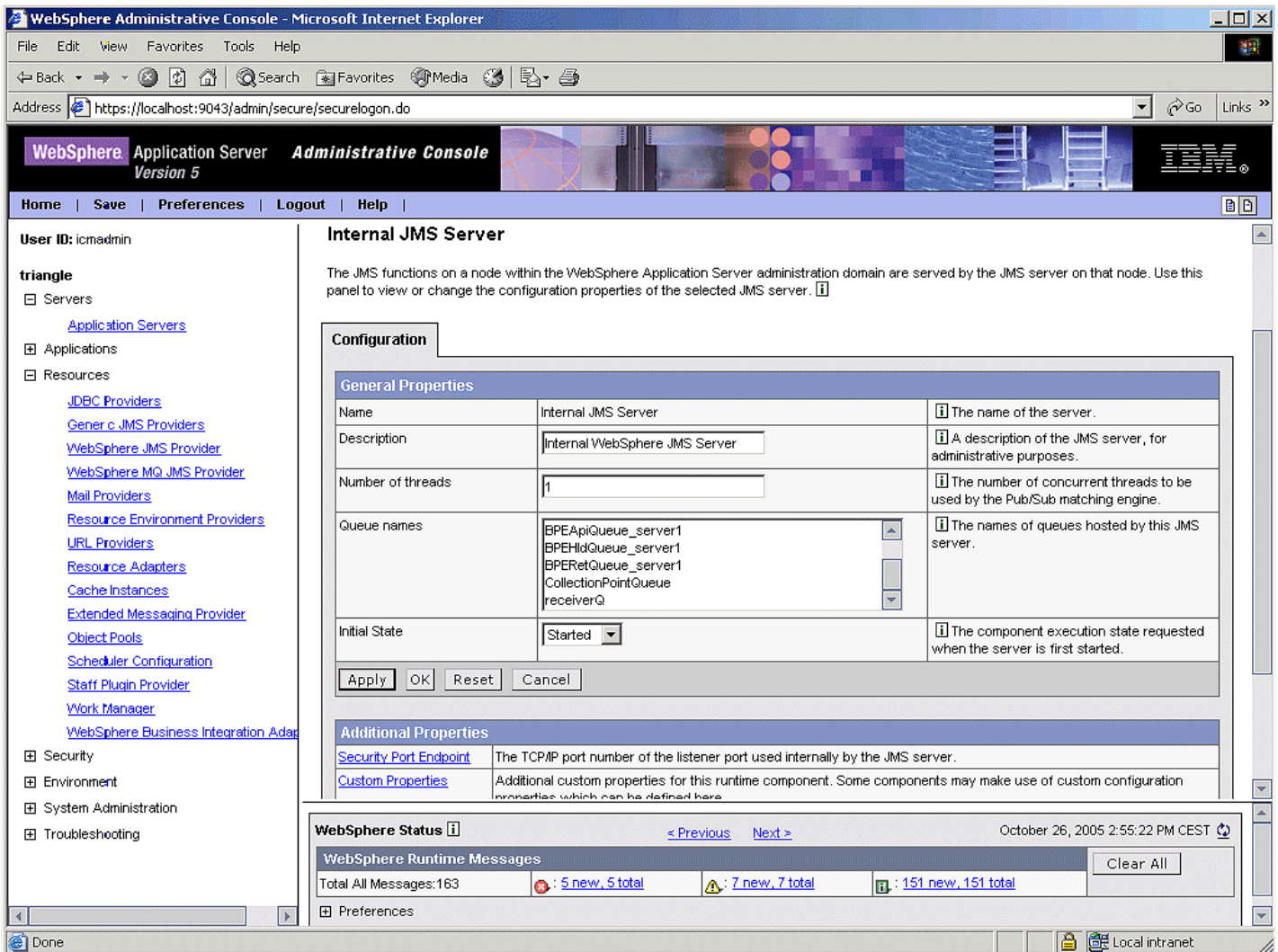


- In the navigation panel select Servers > Application Servers, click **server1** and follow the Additional Properties to **Message Listener Service > Listener Ports**.
- Verify the existence of the three listener ports BPEApiListenerPort, BPEHoldListenerPort, and BPEInternalListenerPort that have been created during process container configuration.
- Click **New** to configure a new listener port for server1 as follows:

| | |
|-------------------------------------|--|
| Name | CollectionPointQListenerPort |
| Description | optionally Listener port for the CM/PC Integration Quick Start |
| Connection factory JNDI Name | jms/CollectionPointQCF |
| Destination JNDI Name | jms/CollectionPointQueue |
- Click **OK** to save the listener port definition.



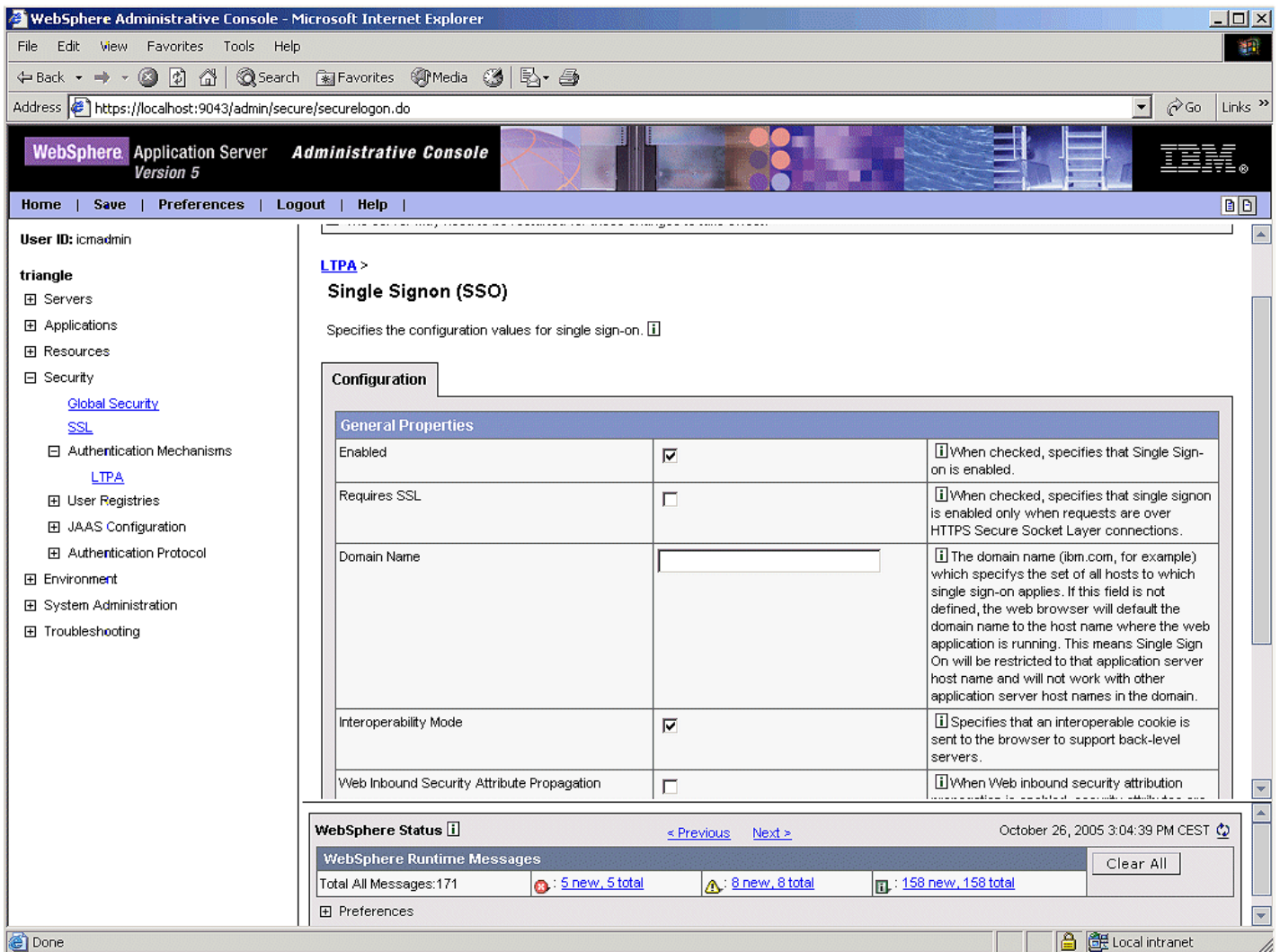
- In the navigation panel select Servers > Application Servers, click **server1** and follow the Additional Properties to **Server Components > JMS Servers**.
- Verify that the **Queue names** field lists the four queues BPEIntQueue_<SERVERNAME>, BPEApiQueue_<SERVERNAME>, BPEHldQueue_<SERVERNAME>, and BPERetQueue_<SERVERNAME> that have been added during process container configuration.
- Add CollectionPointQueue and receiverQ to the list of **Queue names** and click **OK**



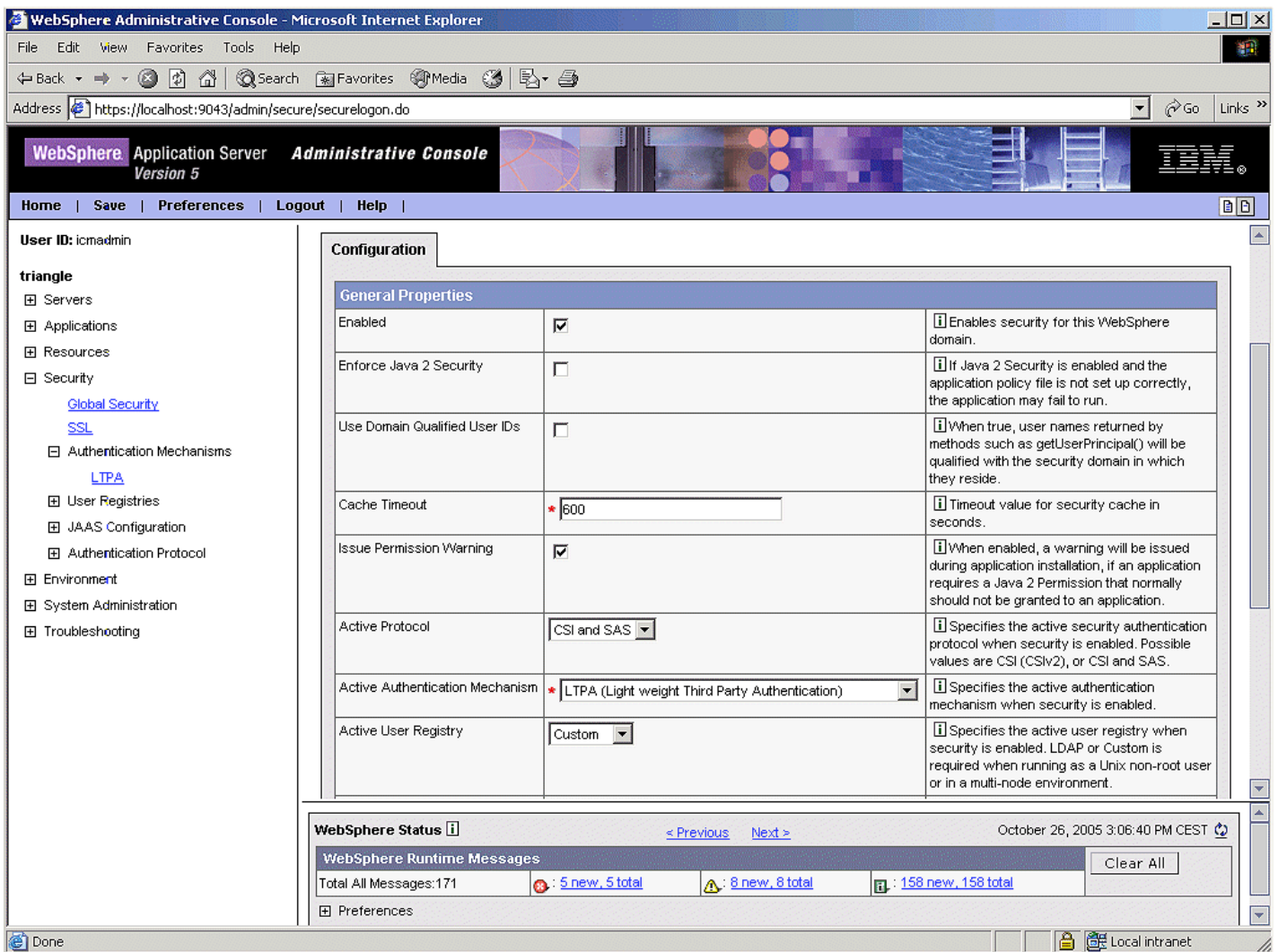
- Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.

5.6 Configure security

- Double-check the security settings in **Security > Global Security** that have been configured during LDAP enablement:
 - The **Enabled** checkmark is checked
 - **Enforce Java 2 Security** is unchecked
 - The **Active User Registry** is set to LDAP
- Double-check the LDAP settings in **Security > User Registries > LDAP**:
 - **Server User ID** and **Server Password** are set to the library server credentials.
 - **Type** is set to Custom.
 - The Hostname and Port are correct.
 - The **Base Distinguished Name** is o=XYZInsurance.
 - Click **Advanced LDAP Settings** and ensure the **UserFilter** is set to `(&uid=%v)(objectclass=person)`.
- In the navigation panel select **Security > Authentication Mechanisms > LTPA**.
- Enter icmadmin's password into the **Password** and **Confirm Password** fields (any other password will do as well).
- Click **Apply**.
- Verify that Single Sign on is enabled by clicking **Single Sign (SSO)** in the Additional Properties section and validate if the **Enabled** check-box is checked.
- In the navigation panel select **Security > Global Security**.
- Ensure the **Enabled** check box is checked and the **Enforce Java 2 Security** check box is unchecked.



- Select LTPA (Light weight Third Party Authentication) from the list of values for the field **Active Authentication Mechanism** and Custom from the list of values for the field **Active User Registry**.
- Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.



- Stop and re-start **server1**.

5.7 Deploying an Integrated Solution on WebSphere Business Integration Server Foundation

Though the title of this section refers to the sample process the same set of steps needs to be performed when installing a custom Content Manager / Process Choreographer Integration solution. Note that connection factory and queue definitions may not be needed if the solution does involve Content Event Handling or the Collection Point Service. In this case the corresponding dependencies need to be disabled in the Integration Toolkit (e.g. the Asynch Bean).

- Start *WebSphere Studio Application Developer Integration Edition* with the workspace created in chapter 1.
- In the **Services** view of the **Business Integration** perspective expand **Deployable Services**, right-click **ClaimsHandlingProjectEJB** and select **Generate > EJB to RDB Mapping**.
- Click **Create a new backend folder** and click **Next**.
- On the page **Create new EJB/RDB Mapping** keep the default **Top down** and click **Next**.
- On the page **Select Top Down Mapping Options** select **DB2 Universal Database V8.1** from the list of values for the **Target Database** field and enter **BPEDB** into the **Database name** field.

EJB to RDB Mapping

Create new EJB/RDB Mapping

Select Top Down Mapping Options:

Target Database:
DB2 Universal Database V8.1

Database name:
BPEDB

Schema name:
NULLID

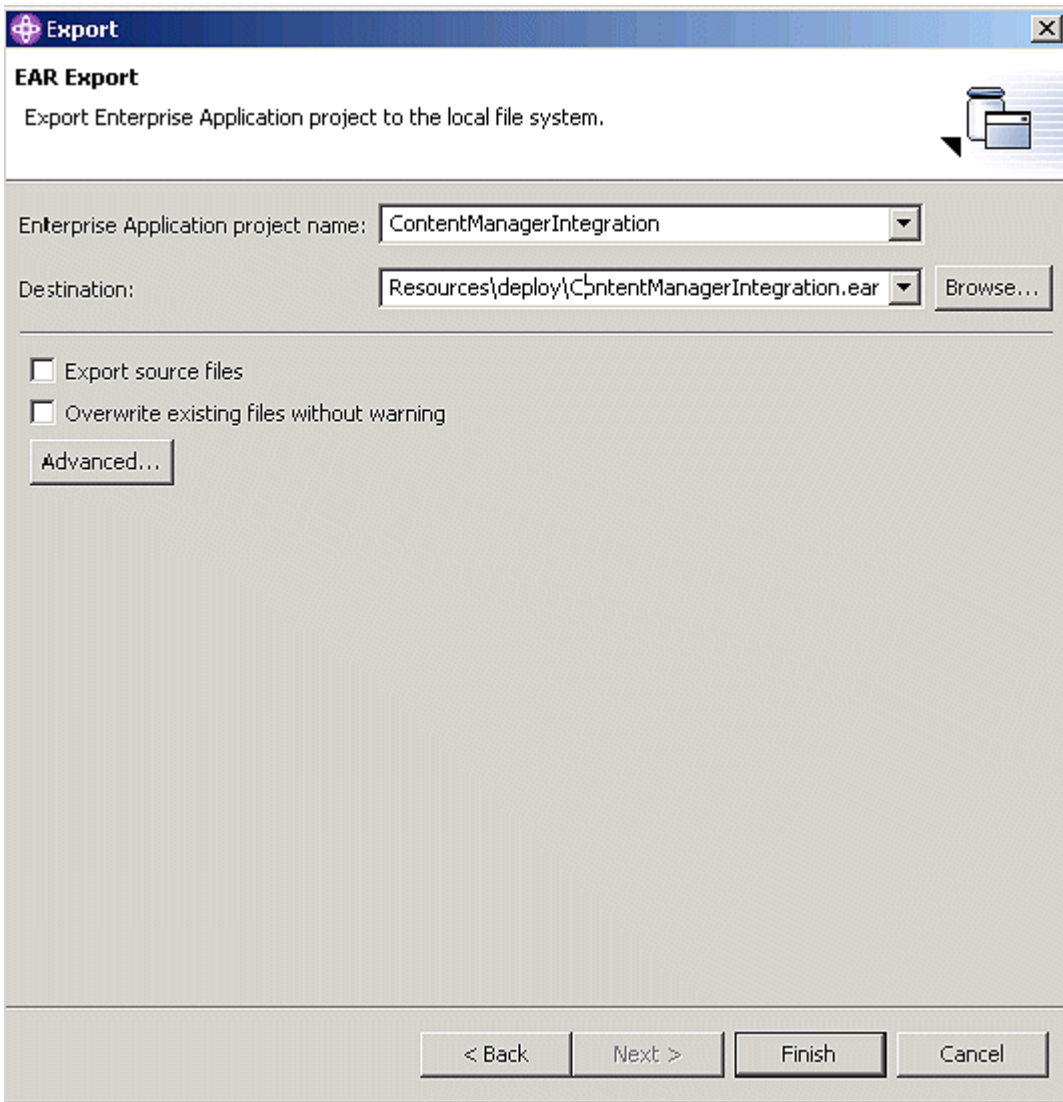
Generate DDL

WebSphere 3.x Compatible

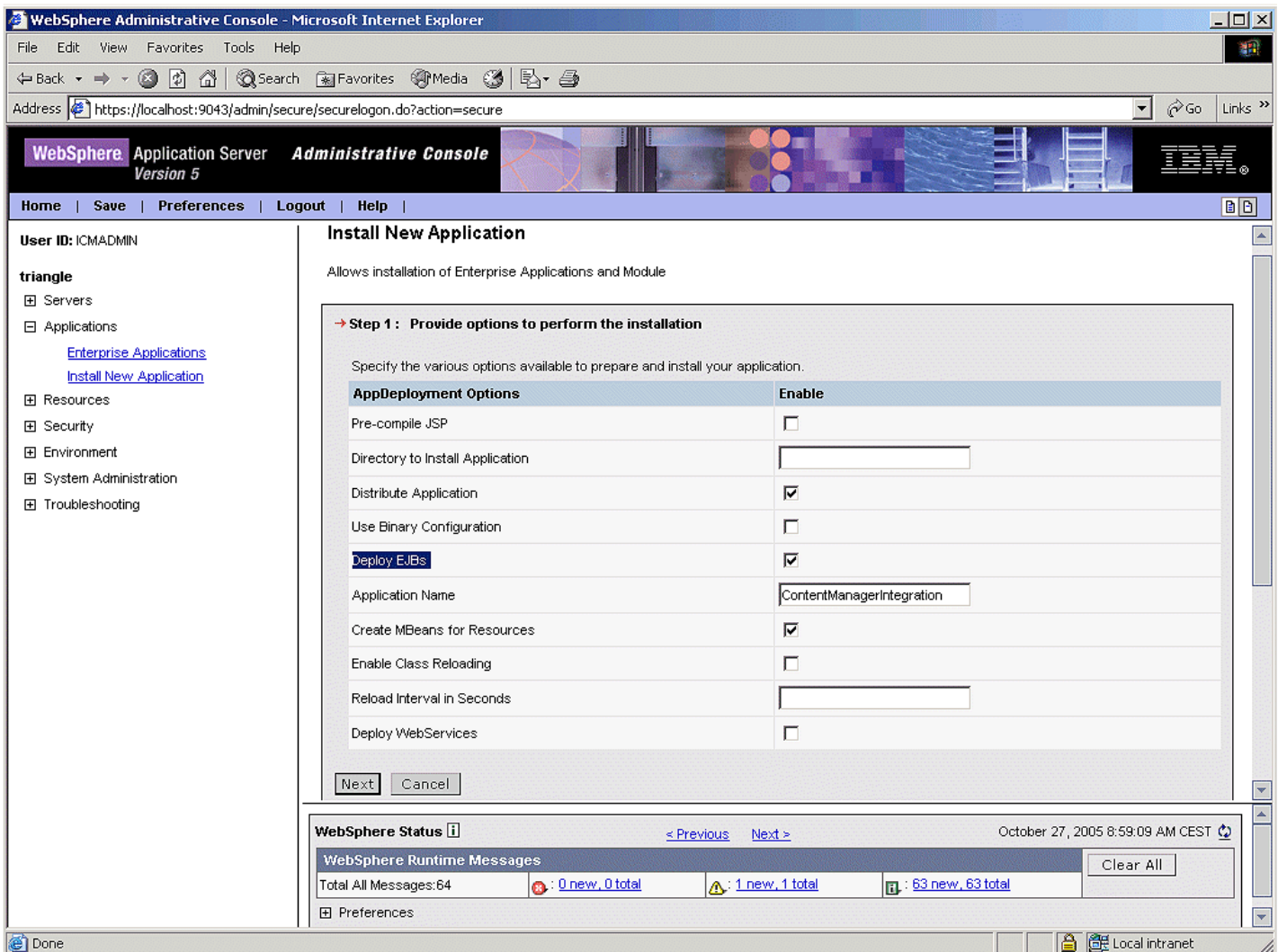
Advanced Options for Inheritance:
Click next to go the Advanced Options Page.

< Back Next > Finish Cancel

- Click **Finish** to create the mapping. The right hand window now displays the database mapping editor with the enterprise beans that are associated with the process. Close this window.
- In the **Services** view expand **Service Projects > ClaimsHandlingProject > com.ibm.bpe.cm.sample**, right-click **ClaimsHandlingProcess.bpel** and select **Enterprise Services > Generate Deploy Code**.
- Click **OK** in the **Generate BPEL Deploy Code** window to initiate the deploy code generation.
- Wait until the progress information window disappears which indicates completion of this step.
- Create a target folder in your file system where the EAR files will be stored.
- On the menu bar select **File > Export**. From the list of export formats select **EAR file**.
- Export the two projects **ClaimsHandlingProjectEAR** and **ContentManagerIntegration** by selecting the project from the list, entering the target folder with the project name followed by **.ear** into the **Destination** field and clicking **Finish**.



- Launch the WebSphere Administrative Console and logon as icmadmin.
- In the navigation panel select Applications > Install New Application.
- Select **Browse** to locate ContentManagerIntegration.ear and click **Next**.
- On the panel for default binding and mapping generation click **Next**.
- On the panel **Step 1: Provide options to perform the installation** check **Deploy EJBs** and click **Next**



- On the panel **Step 2: Provide options to perform the EJB Deploy** verify that the value of the field **Deploy EJBs Option - Database Type** is set to `DB2UDB_V81`.
- Go to panel **Step 13: Summary**, verify the settings, and click **Finish**.

WebSphere Administrative Console - Microsoft Internet Explorer

Address: https://localhost:9043/admin/secure/securelogin.do?action=secure

WebSphere Application Server Administrative Console Version 5

Home | Save | Preferences | Logout | Help

User ID: ICMADMIN

triangle

- Servers
- Applications
 - Enterprise Applications
 - Install New Application
- Resources
- Security
- Environment
- System Administration
- Troubleshooting

Step 13: Summary

Summary of Install Options

| Options | Values |
|--------------------------------------|----------------------------|
| Deploy EJBs Option - RMIC | |
| Deploy EJBs Option - Classpath | |
| Distribute Application | Yes |
| Use Binary Configuration | No |
| Cell/Node/Server | Click here |
| Enable Class Reloading | No |
| Create MBeans for Resources | Yes |
| Deploy EJBs | Yes |
| Reload Interval in Seconds | |
| Application Name: | ContentManagerIntegration |
| Directory to Install Application | |
| Deploy EJBs Option - Database Type | DB2UDB_V81 |
| Pre-compile JSP | No |
| Application Name | ContentManagerIntegration |
| Deploy EJBs Option - Database Schema | |
| Deploy WebServices | No |

Previous Finish Cancel

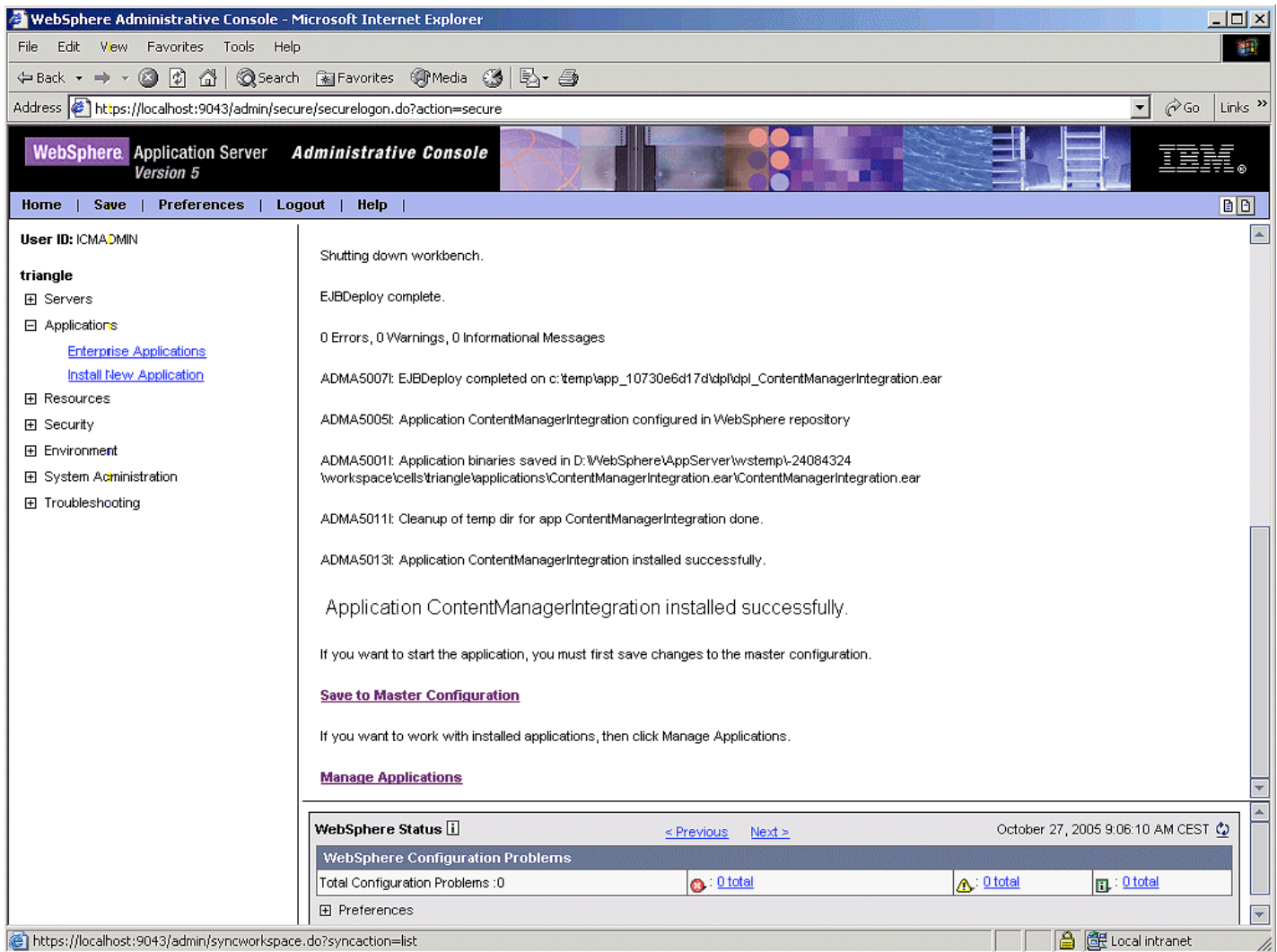
WebSphere Status < Previous Next > October 27, 2005 9:05:10 AM CEST

WebSphere Runtime Messages

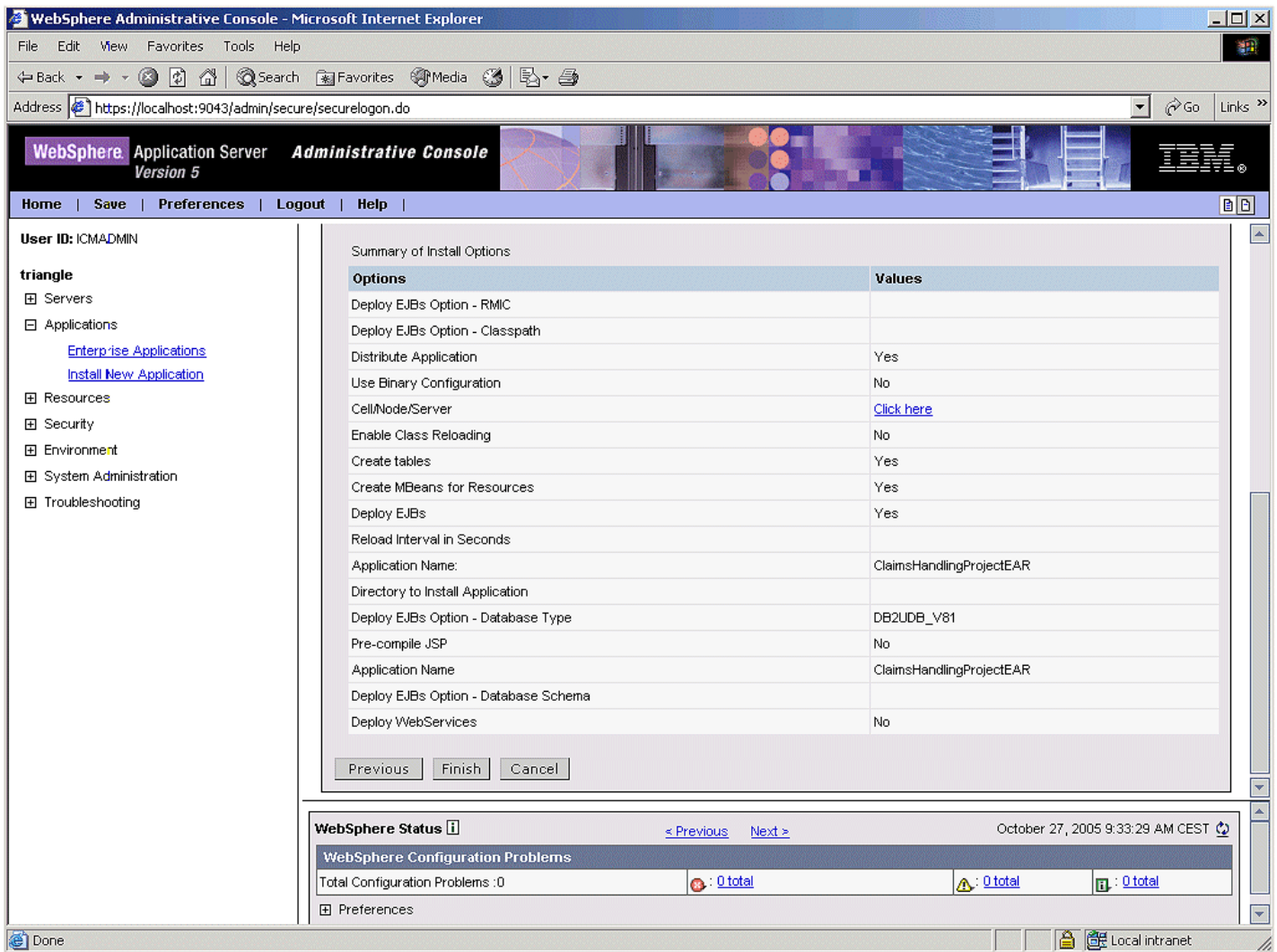
Total All Messages: 72 : 0 new, 0 total : 2 new, 2 total : 70 new, 70 total

Preferences

- Wait for the message Application ContentManagerIntegration installed successfully (you may need to scroll down the panel content to see it).
- Select the link **Save to master configuration** and click **Save** to make the update permanent.



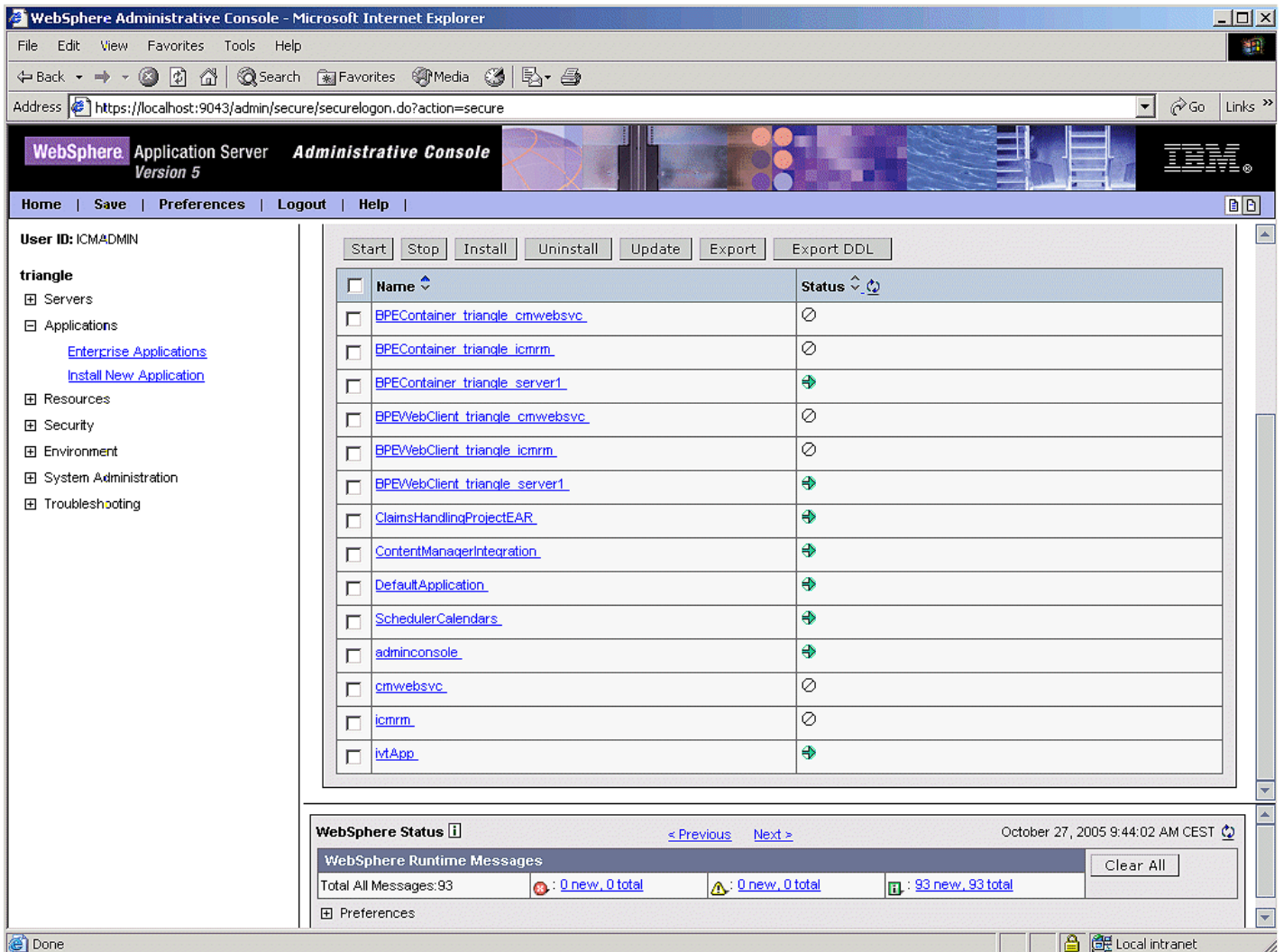
- In the navigation panel select Applications > Enterprise Applications, check the checkmark left of **ContentManagerIntegration** and click **Start**. The Status of this application should then turn into a green arrow indicating that the application has been started successfully.
- In the navigation panel select Applications > Install New Application.
- Select **Browse** to locate ClaimsHandlingProjectEAR and click **Next**.
- On the panel for default binding and mapping generation click **Next**.
- On the panel **Step 1: Provide options to perform the installation** check **Deploy EJBs** and click **Next**.
- On the panel **Step 2: Provide options to perform the EJB Deploy** verify that the value of the field **Deploy EJBs Option - Database Type** is set to DB2UDB_V81 and click **Next**.
- On the panel **Step 3: Select current backend ID** verify that the value of the column **CurrentBackendId** is set to DB2UDBNT_V8_1 and click **Next**.
- Go to panel **Step 13: Automatically create database tables for Business Process entity beans** and check **Enable**.
- Go to panel **Step 14: Summary**, verify the settings, and click **Finish**.



- Wait for the message Application ClaimsHandlingProjectEAR installed successfully (you may need to scroll down the panel content to see it).
- Select the link **Save to master configuration** and click **Save** to make the update permanent.

5.8 Install the runtime files, start the server and run the sample

- Copy the files `bpecm.jar` and `bpecmutil.jar` from `<WSADIE_HOME>\runtimes\ee_v51\lib` to `<WAS_HOME>\lib`.
- Copy the file `bpecm.properties` from `<WSADIE_HOME>\runtimes\ee_v51\properties` to `<WAS_HOME>\properties`.
- In the navigation panel of the WAS administrative console select `Applications > Enterprise Applications`, check the checkmark left of **ClaimsHandlingProjectEAR** and click **Start**. The Status of this application should then turn into a green arrow indicating that the application has been started successfully.



- Now you should be able to access the application through your browser.

6. Hints and Troubleshooting

Traps when working with the sample

- When opening a new case the claim form needs to be the first document that is imported and it needs to have claim number that has not yet been used. In case of doubt you can use the *Client for Windows* to find all instances of `XYZ_ClaimFolder` and check their values for `XYZ_ClaimNumber`.
- If an adjuster or police report with a new claim number is stored in DB2 Content Manager this triggers the creation of a corresponding folder and process instance. Since neither of these items has a policy insurance number assigned, the 'RetrievePolicy' service fails but the process continues as usual.
- If two claim forms with the same claim number exist, creating the second one causes an error message to show up in the console output since the 'RetrievePolicy' services tries to add the insurance policy to the folder. The document can not be added to a folder twice.

Resolve BPEL references by binding services to port types

If a project has been deleted and needs to be re-installed the references may need to be re-bound as follows.

- In the **Generate BPEL Deploy Code** window select `Referenced Partners > CollectionPoint` in the navigation pane on the left, click **Browse** on the right hand side, locate the file `ContentManagerIntegrationJar\com\ibm\bpe\cm\CollectionPointJMSService.wsdl` in your workspace and click **OK** to link the port type to this service.
- Select `Referenced Partners > FolderManagement` in the navigation pane on the left, click **Browse** on the right hand side, locate the file `ContentManagerIntegrationJar\com\ibm\bpe\cm\FolderManagementJavaService.wsdl` in your workspace and click **OK** to link the port type to this service.

Clean up the process environment

- Stop the test server.

- Run the following advanced query in the *DB2 Content Manager Client for Windows*: `/*[<Claim Number (Content Manager V8.1 Sample Attribute)> LIKE "3-%"]` to identify all items created when running the Quick Start Client.
- Delete all items of the result list.
- Connect to `icmnlbdb` as `icmadmin` and drop the three tables `bpecontentevents`, `bpecollectionpoints`, `bpecollectionpointtitemtypes`.
- Restart the test server. This re-creates the three tables.

Upgrading the content viewer applet to fix pack 1 or (re-)creating the content viewer applet

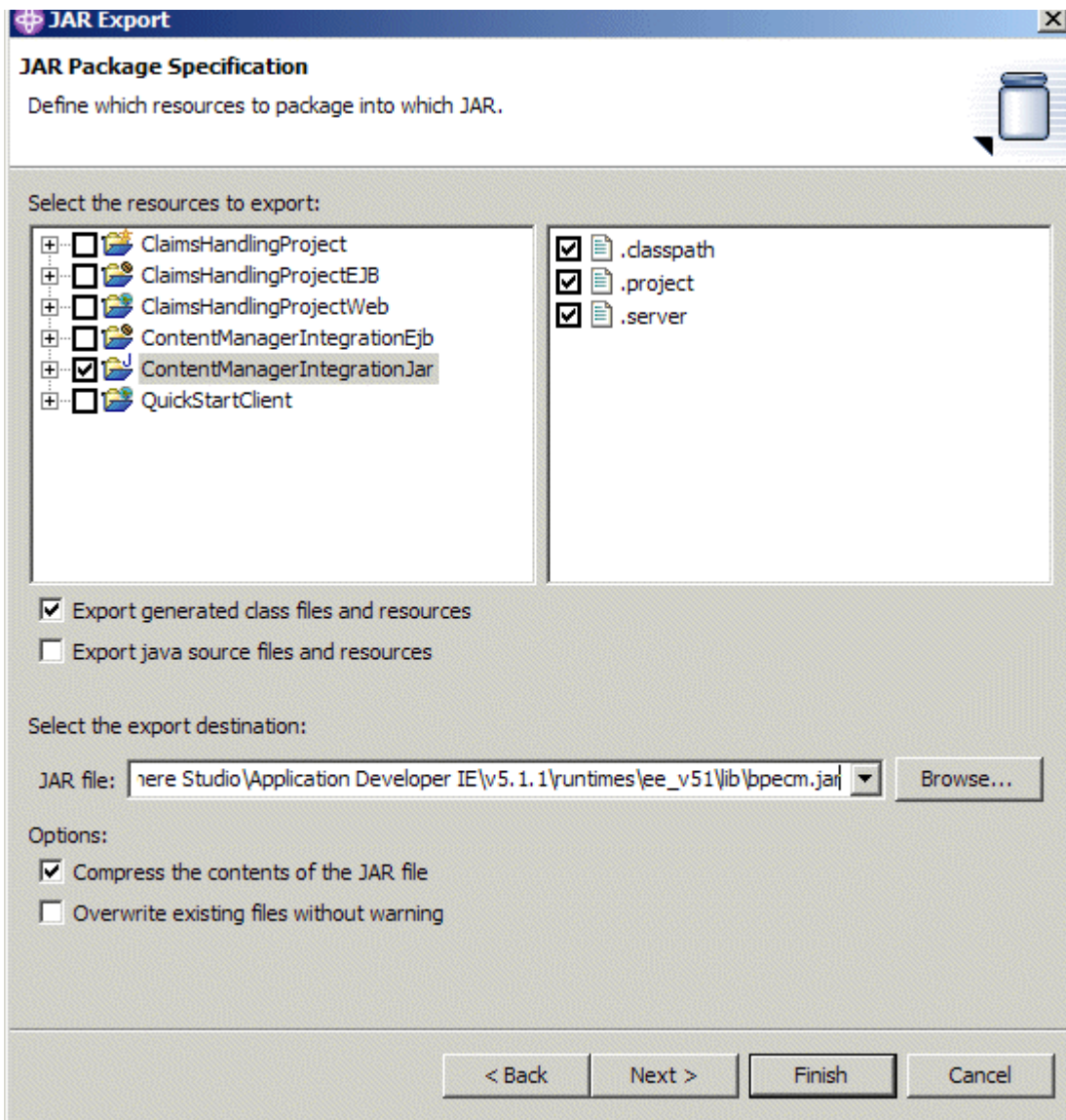
The Quick Start Client uses a modified version of the Content Manager viewer applet to display document content. This applet and the required resources need to be placed into a specific directory where the files are available for download by the client. This step copies the required resources from the Content Manager install directory and (re-)creates the JAR file for the modified viewer applet. This step should be performed when Fix Pack 1 is used or when the code in `TViewerApplet.java` has been changed (See [DB2 Content Manager Enterprise Edition \(multiplatform\) Support](#) for Fix Pack downloads, etc.).

Important: (Re-)creating the applet viewer requires *DB2 Content Manager Enterprise Edition V8.3 Fix Pack 1* to be installed. If Fix Pack 1 is installed we recommend to (re-)create the viewer applet to take advantage of the latest changes in the viewer code.

- Switch to the **Services** view of the business integration perspective.
- Expand **Deployable Services > QuickStartClient > WebContent**.
- Perform the following steps to import `cmbview81.jar` into this folder.
 - From the workspace menu select **File > Import**. The 'Import' window opens.
 - Select **File system** and click **Next >**.
 - Click the **Browse** button to the right of the **From directory:** field, locate the folder `<IBMCMROOT>\lib`, and click **OK**.
 - In the field to the right select `cmbview81.jar`.
 - Enter `QuickStartClient/WebContent/appletViewer` in the **Into folder:** field.
 - Click **Finish**.
- In the 'Services' view right-click `QuickStartClient/WebContent/appletViewer/cmbview81.jar` and select **Refresh**.
- Switch to the 'Package Explorer' view and perform the following steps to create the JAR file for the viewer applet. You need to re-run these steps if you have modified the source code of the viewer applet:
 - Right-click `ContentViewerApplet` and select **Export**. The 'Export' window opens.
 - Select **JAR file** and click **Next>** which switches to the 'JAR Package Specification' dialogue.
 - In the 'Select the export destination' field click **Browse** and locate the folder `<WORKSPACE_FOLDER>\QuickStartClient\WebContent\appletViewer`. Type `appletViewer.jar` into the **File name:** field and click **Save**.
 - Check 'Overwrite existing files without warning' and click **Finish**.
- In the 'Package Explorer' view right-click `QuickStartClient/WebContent/appletViewer/appletViewer.jar` and select **Refresh**.

Creating and deploying a new Integration Toolkit library

- Stop the test server.
- Back up the current version of `<WSADIE_ROOT>/runtimes/ee_v51/lib/bpecm.jar`.
- Click `ContentManagerIntegrationJar` in the navigation pane and select **Project > Rebuild All** on the menu to make sure all files of the project are up-to-date.
- Right-click `ContentManagerIntegrationJar` in the navigation pane and select **Export**. The **Export** window opens.
- Select **Jar file** and click **Next >**. On the **Jar Package Specification** window verify that it looks as follows:



- Click browse to locate `<WSADIE_ROOT>/runtimes/ee_v51/lib` and enter the file name `bpecm.jar`.
- When asked if you want to overwrite the existing file click **Yes**.
- Upon completion, a JAR Export window might pop up that says JAR export finished with warnings. See details for additional information. The details say `/ContentManagerIntegration/META-INF/MANIFEST.MF` was replaced by the generated `MANIFEST.MF` and is no longer in the JAR. This can safely be ignored. Click **OK** to close the **JAR export** window.

7. Further Reading

Product documentation

- [DB2 Content Manager V8.3 Infocenter](#) on the Web
- [WebSphere Application Server Version 5.1.x information center](#) on the Web (also covering WebSphere Business Integration Server Foundation, Version 5.1.x)

Background information and documents on specific topics

- [WebSphere Business Integration Server Foundation V5.1 Handbook \(IBM redbook SG24-6318\)](#)
- [BPEL4WS Business Processes with WebSphere Business Integration: Understanding, Modelling, Migrating \(IBM redbook SG24-6381\)](#)
- [WebSphere Business Integration Server Foundation Using the programming API and the Common Event Infrastructure \(IBM Redpaper REDP-3915\)](#)

Product support Web sites

- [WebSphere Studio Application Developer Integration Edition Support](#)
- [WebSphere Business Integration Server Foundation Support](#)
- [DB2 Content Manager Enterprise Edition \(multiplatform\) Support](#)

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

| | | |
|-----------|---------|------------------------|
| IBM | i5/OS | z/OS |
| Redbooks | AIX | Cloudscape |
| DB2 | DB2 UDB | DB2 Universal Database |
| WebSphere | Lotus | Tivoli |

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.