DB2 Content Manager / WebSphere Process Server Integration Quick Start

Revision2

This tutorial describes the integration of *DB2 Content Manager Enterprise Edition Version 8.3* with *WebSphere Process Server V6.0.1*. This document consists of the the following chapters:

1. Introduction

DB2 Content Manager Enterprise Edition V8.3 includes a high-performance document routing engine to direct documents and folders of documents from one user to another. For complex process automation requirements such as those that involve human tasks outside the document environment or integrations with a variety of applications, systems and services, *WebSphere Process Server* is the process management environment of choice.

WebSphere Process Server uses a service-component architecture to deliver one programming model to connect and use existing IT resources to solve business problems. Using simplified tooling to describe and create the solution, components can be brought together to create a managed business process. Processes can be based on automated steps that together constitute a single transaction or they can involve human interaction and can run for days, weeks or months. Processes types vary greatly, ranging from Web services or Web page navigation to business transaction support.

The DB2 Content Manager / WebSphere Process Server (CM/WPS) Integration Quick Start provides some key ingredients for the creation of content-centric processes. The term "content-centric" refers to a process in which a content object, typically a document or image, stored in the content management system plays an important role. Examples are review processes such as those involved in publishing, invoice processing, or claims handling. Content-centric processing is also seen in Business To Business (B2B) scenarios where electronic documents are exchanged between different companies.

Human activities and automated steps that modify the content object are key elements of content-centric processes. The content object is usually a folder and is said to be *routed through the process or (work)flow*. Modifying it can mean a number of things such as adding documents to or removing documents from the folder, changing attribute values on documents, modifying access rights, adding annotations, signatures, or watermarks, etc.

The DB2 Content Manager / WebSphere Process Server Integration Quick Start provides best practices and pre-assembled building blocks that support the creation and management of content-centric processes based on the WebSphere Process Server and DB2 Content Manager environments. Typical interaction patterns between the process engine and the content management system are demonstrated through code samples and comprehensive documentation.

This Quick Start builds on previous versions which showed interaction between DB2 Content Manager and the Business Process Choreographer component of WebSphere Business Integration Server Foundation.

1.1 What's New in Revision 2

The main topic of Revision 2 is support of WebSphere Process Server V6. Revision 2 of the 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 and EJBs) that implement the following interaction patterns between the content management and process management system:

- **Content Event Handling** ⇒ Item creation or deletion events in DB2 Content Manager trigger corresponding actions in Process Server.

- **Collection Point** ⇒ A point where a process instance waits 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 Server and associated content attribute values from DB2 Content Manager.
- **Common Staff Repository** ⇒ Map DB2 Content Manager user and group definitions to the Process Server 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 can involve legacy applications or B2B transactions.

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

The following statements apply to the source code of the DB2 Content Manager / WebSphere Process Server Integration Quick Start:

The integration toolkit requires a DB2 Universal Database-based library server and it does not support 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 DB2 Content Manager library server needs to be based on a *DB2 Universal Database V8*. For details on the required DB2 UDB fixpack levels see the DB2 Content Manager product documentation. The integration toolkit only supports a DB2 Universal Database-based library server. Note that Fixpack 2 of DB2 Content Manager V8.3 is a prerequisite of the CM/WPS Integration Quick Start if you want to use the Custom CM-based user registry. Where DB2 Content Manager V8.3 is mentioned in the following sections this alway implies at least Fixpack 2.

The following system configuration is required for the development and test environment (chapter 2):

a. Development workstation A hosts:
   - *WebSphere Integration Developer V6.0.1*
   - *WebSphere Process Server V6.0.1 with JDK 1.4.2 Service Release 4-1 (SR4-1)* - installed as integrated test environment of WebSphere Integration Developer or separately (recommended)
   - *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 (Fix Pack 2)*

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

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

`<IBMCMROOT>`	=	DB2 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\sqllib` or `/home/db2inst1/sqllib`)
`<WID_HOME>`	=	WebSphere Integration Developer install folder (e.g. `c:\Program Files\IBM\Websphere\ID\6.0`)
`<WPS_HOME>`	=	WebSphere Process Server install folder (e.g. `c:\Program Files\IBM\Websphere\ProcessServer`)
`<WPS_PROFILE_HOME>`	=	folder of the WebSphere Integration Developer profile to be used on the development workstation (e.g. `c:\Program Files\IBM\Websphere\IntegrationDeveloper\profiles\ProcSrv01`)

Note that we are using the default `icmadmin` to refer to the ID of the library server administrator. If the ID of your library server administrator is different, replace all occurences of `icmadmin` with the ID used on your system.

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*. *4. Developing a Custom DB2 Content Manager / WebSphere Process Server Integration Solution* introduces the individual elements of the Integration Toolkit and describes the steps necessary when creating a custom CM/WPS integration solution based on the Integration Toolkit and Quick Start Client *5. Invoking DB2 Content Manager Web Services from within a process* outlines how Content Manager Web Services can be incorporated into a process. *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 contains 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.

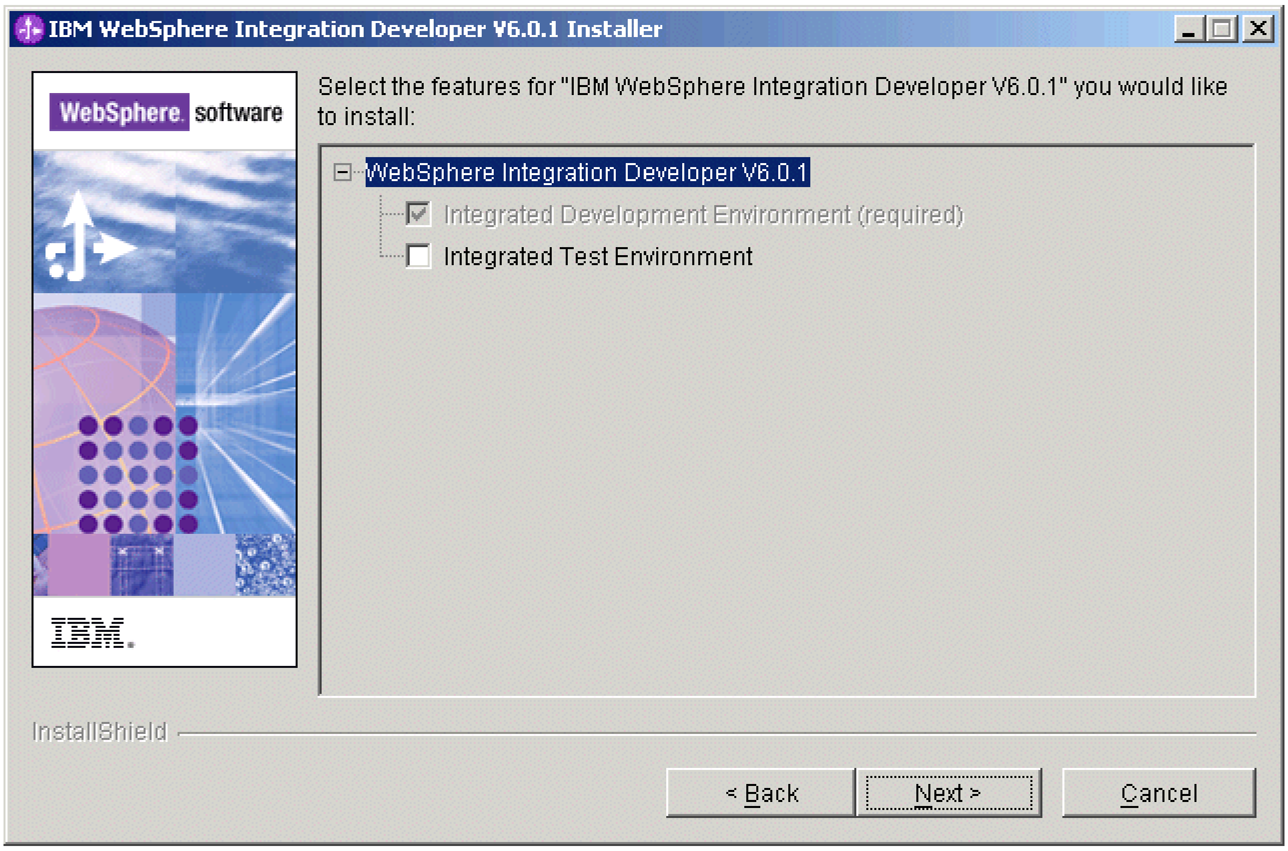
Installation hints for Information Integrator for Content

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 `<IBMCMROOT>`.
- **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 DB2 Content Manager server.

Installation hints for the test environment

You can either install with the WebSphere Integration Developer V6.0.1 with a test environment or install WebSphere Process Server V6.0.1 separately. We recommend a separate installation of WebSphere Process Server as this offers more explicit control over port and user settings. If you want to install WebSphere Process Server with the WebSphere Integration Developer product you need to check **Integrated Test Environment** during installation. In this case a default profile is created automatically with `<WPS_PROFILE_HOME>` set to `<WID_HOME>\pf\wps` .


IBM WebSphere Integration Developer V6.0.1 Installer

Select the features for "IBM WebSphere Integration Developer V6.0.1" you would like to install:

- WebSphere Integration Developer V6.0.1
  - [✓] Integrated Development Environment (required)
  - [ ] Integrated Test Environment

If you decide to install WebSphere Process Server separately, you need to create a profile to be used as test environment based on the following steps:

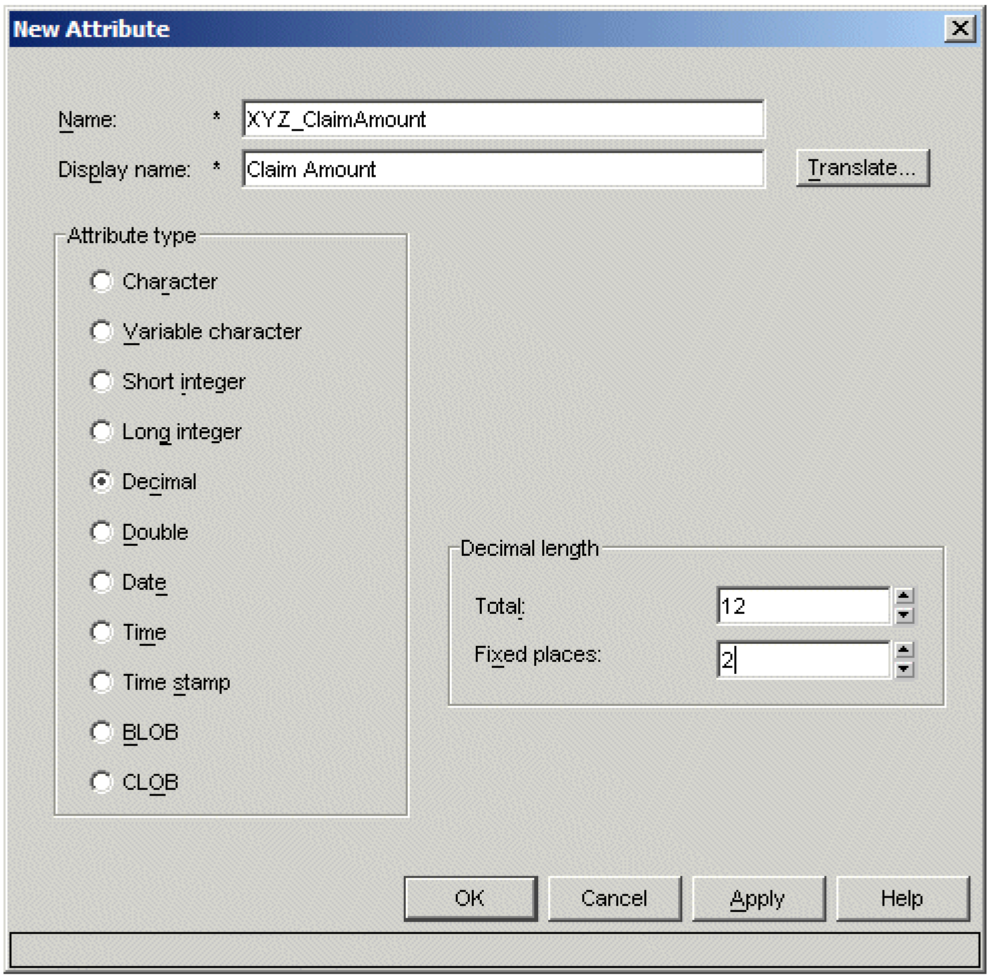
1. Start the Profile Creation Wizard
2. Select a profile name. The suggested name `ProcSrv01` may be OK. You may want to make this your default profile.

3. Specify a profile directory. We recommend that you make the path to this directory as short as possible especially on Windows systems where the operating system limit of 256 characters is exceeded easily.
4. Verify if the node name and fully specified host name are correct. They should usually be OK.
5. Carefully investigate the port assignments on the 'Port valua assignment' page. Make sure there is no conflict with other applications on the server. Potential candidates might be the (probably WebSphere Application Server V5-based) servers that are hosting the administrative console (server1) resource manager (icmrm) or the Content Manager Web services (cmwebsvc). If necessary, modify the port assignments so there are no conflicts. Make notices of the port assignments for HTTP transport and SOAP communication as these are needed later on.
6. It is recommended to uncheck 'Run the WebSphere Process Server process as a Windows service'. The process can be started manually as described in section 6.2.
7. Check 'Configure the Service Integration Bus in secured mode' and enter `icmadmin` as user ID and the corresponding password.
8. On the 'Common Event Infrastructure Configuration' page enter `icmadmin`, the corresponding password, `server1` as the server name, and `Cloudscape V5.1` as the database product.
9. On the 'Business Process Choreographer Configuration' page select **Configure a sample Business Process Choreographer** and enter `icmadmin` with the corresponding password. As the name of the administrative group enter the administrative group to which `icmadmin` belongs. On a Windows system this is usually `Administrators`.
10. Skip the 'Application Scheduler configuration' page.
11. On the 'Database Configuration' page select **Create new (local) database** and accept the default settings for the database product `(Cloudscape)` and name `(WPRCSDB)`.
12. Double-check the summary, click **Next** and wait until the wizard signals completion.

2.2 Preparing the DB2 Content Manager Sample Data for use with the Sample Process

The Quick Start sample process is based on a modified version of the DB2 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 and '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).

1. Ensure that the resource manager application is started. This can be verified by running `serverStatus.bat -all` in the `bin` folder of the WebSphere Application Server installation that hosts the resource manager. Run `startServer.bat icmrm` to start the resource manager if it is not yet running.
2. Start the DB2 Content Manager First Steps program.
3. If you have worked with the sample data before, click **Unload Sample Data** to ensure all data is in a clean state.
4. Click **Load Sample Data**.
5. Wait until the sample data creation completes.
6. Click **Work with Sample Data** to start the DB2 Content Manager system administration client.
7. Log on to the system administration client with your library server administrator userid and password.
8. Select **Data Modeling**, right-click **Attributes**, select **New** to define a new attribute `XYZ_ClaimAmount` with the properties shown below:


New Attribute

Name: * XYZ_ClaimAmount

Display name: * Claim Amount

Attribute type

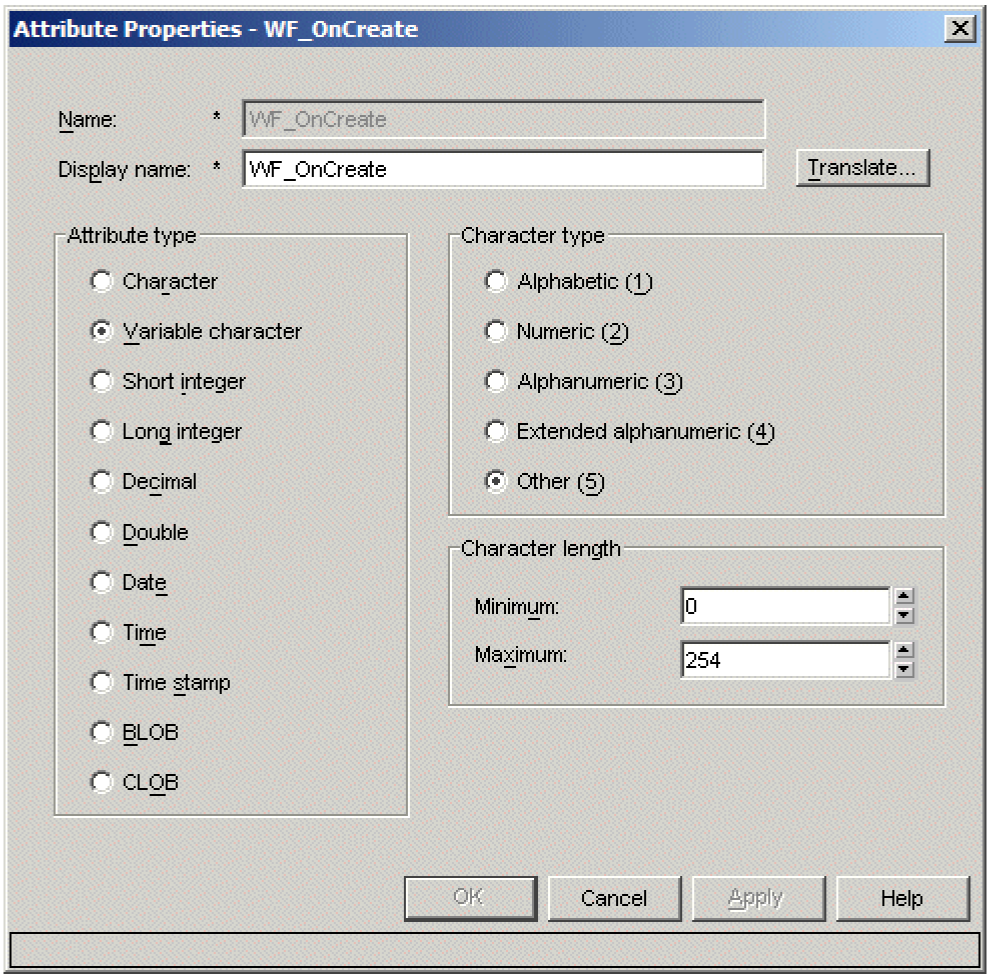
( ) Character
( ) Variable character
( ) Short integer
( ) Long integer
(●) Decimal
( ) Double
( ) Date
( ) Time
( ) Time stamp
( ) BLOB
( ) CLOB

Decimal length

Total: 12

Fixed places: 2

9. Click **OK** to store the new attribute.
10. 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:



Name: * WF_OnCreate

Display name: * WF_OnCreate

Attribute type

- ( ) Character
- [✓ Variable character]
- ( ) Short integer
- ( ) Long integer
- ( ) Decimal
- ( ) Double
- ( ) Date
- ( ) Time
- ( ) Time stamp
- ( ) BLOB
- ( ) CLOB

Character type

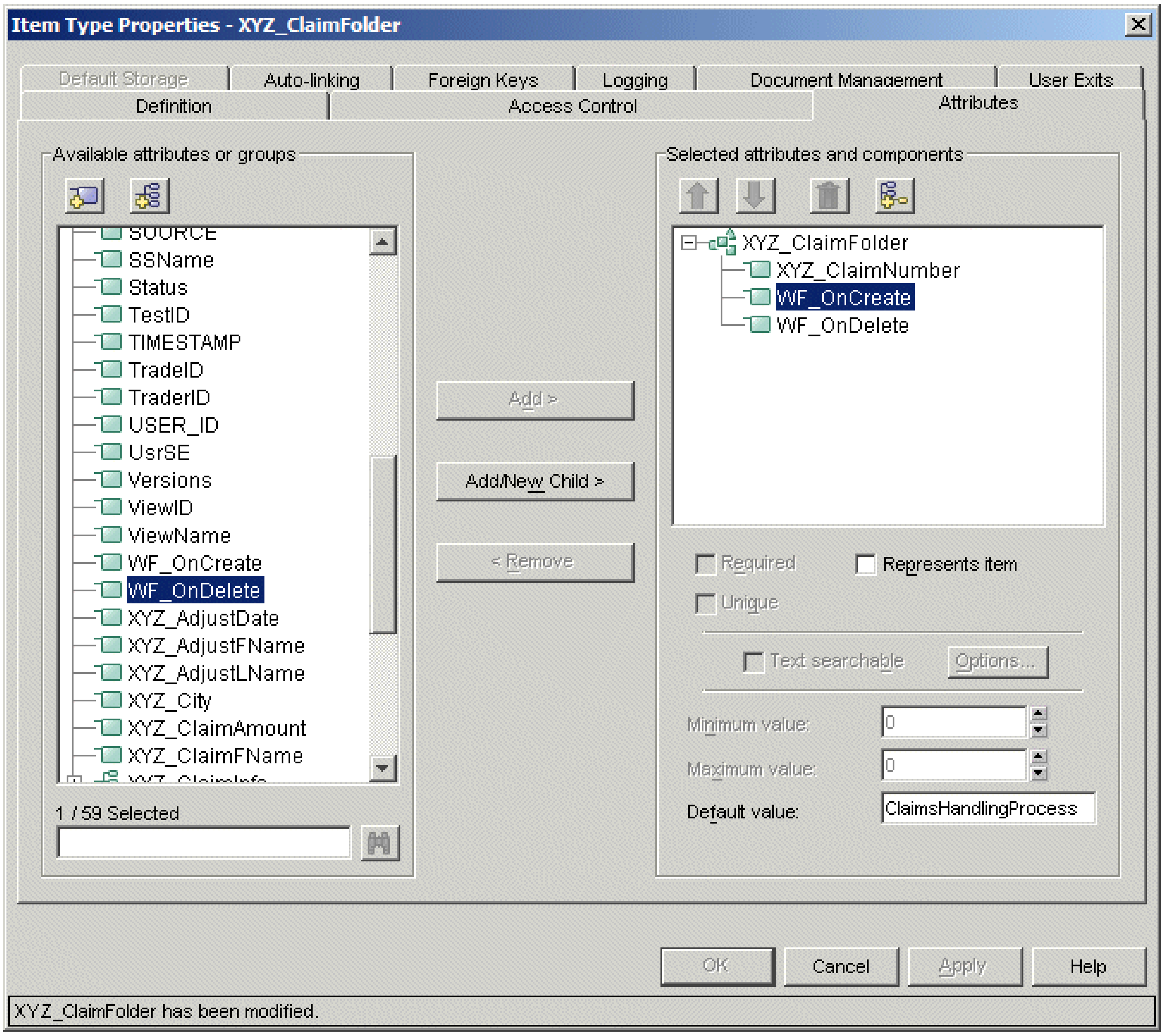
- ( ) Alphabetic (1)
- ( ) Numeric (2)
- ( ) Alphanumeric (3)
- ( ) Extended alphanumeric (4)
- [✓ Other (5)]

Character length

Minimum: 0

Maximum: 254

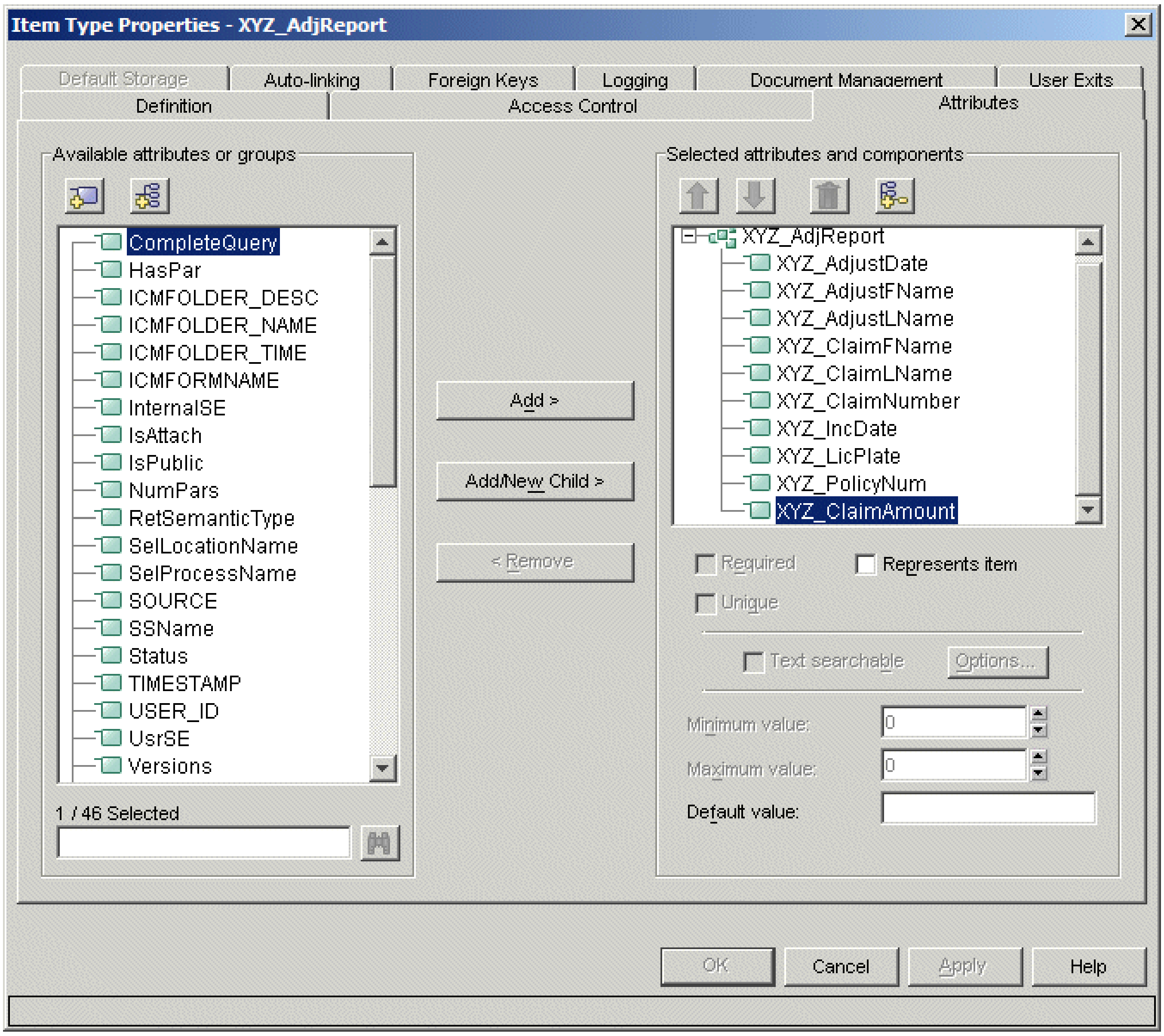
11. Repeat the previous step to define a new attribute `WF_OnDelete` with the same characteristics as `WF_OnCreate`.
12. In the Data Modeling view, expand **Item Types**, right-click on `XYZ_ClaimFolder`, right-click **Properties**, open the **Attributes** tab and add two new attributes: `WF_OnCreate`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



Minimum value: 0
Maximum value: 0
Default value: ClaimsHandlingProcess

XYZ_ClaimFolder has been modified.

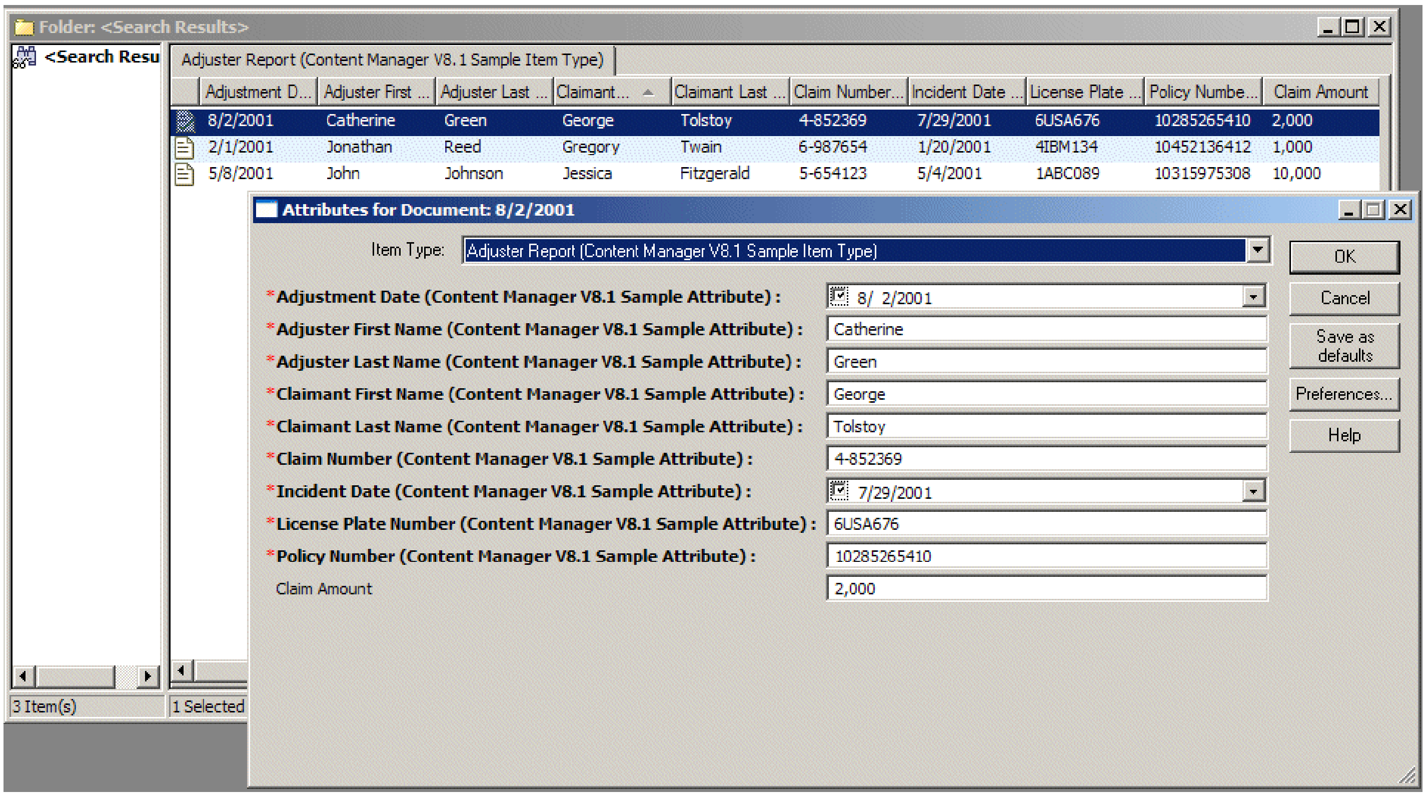
13. Right-click `XYZ_AdjReport` and select **Properties**.
14. On the definition tab, change the **New version policy for attributes** to **Never create**.
15. Switch to the **Attributes** tab and add the attribute `XYZ_ClaimAmount` to this item type, as shown below:



16. Close the System Administration Client.
17. Close the First Steps program.

Entering values for the previously created attributes

1. Start the *DB2 Content Manager Client for Windows* and log on as library server administrator (icmadmin). Note that if the *DB2 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.
2. Select **Search > Basic** and locate all instances of item type `Adjuster Report (Content Manager v8.1 Sample Item Type)`.
3. 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.


Folder: <Search Results>

<Search Resu

Adjuster Report (Content Manager V8.1 Sample Item Type)

Adjustment D...	Adjuster First ...	Adjuster Last ...	Claimant...	Claimant Last ...	Claim Number...	Incident Date ...	License Plate ...	Policy Numbe...	Claim Amount
8/2/2001	Catherine	Green	George	Tolstoy	4-852369	7/29/2001	6USA676	10285265410	2,000
2/1/2001	Jonathan	Reed	Gregory	Twain	6-987654	1/20/2001	4IBM134	10452136412	1,000
5/8/2001	John	Johnson	Jessica	Fitzgerald	5-654123	5/4/2001	1ABC089	10315975308	10,000

3 Item(s)

1 Selected

Attributes for Document: 8/2/2001

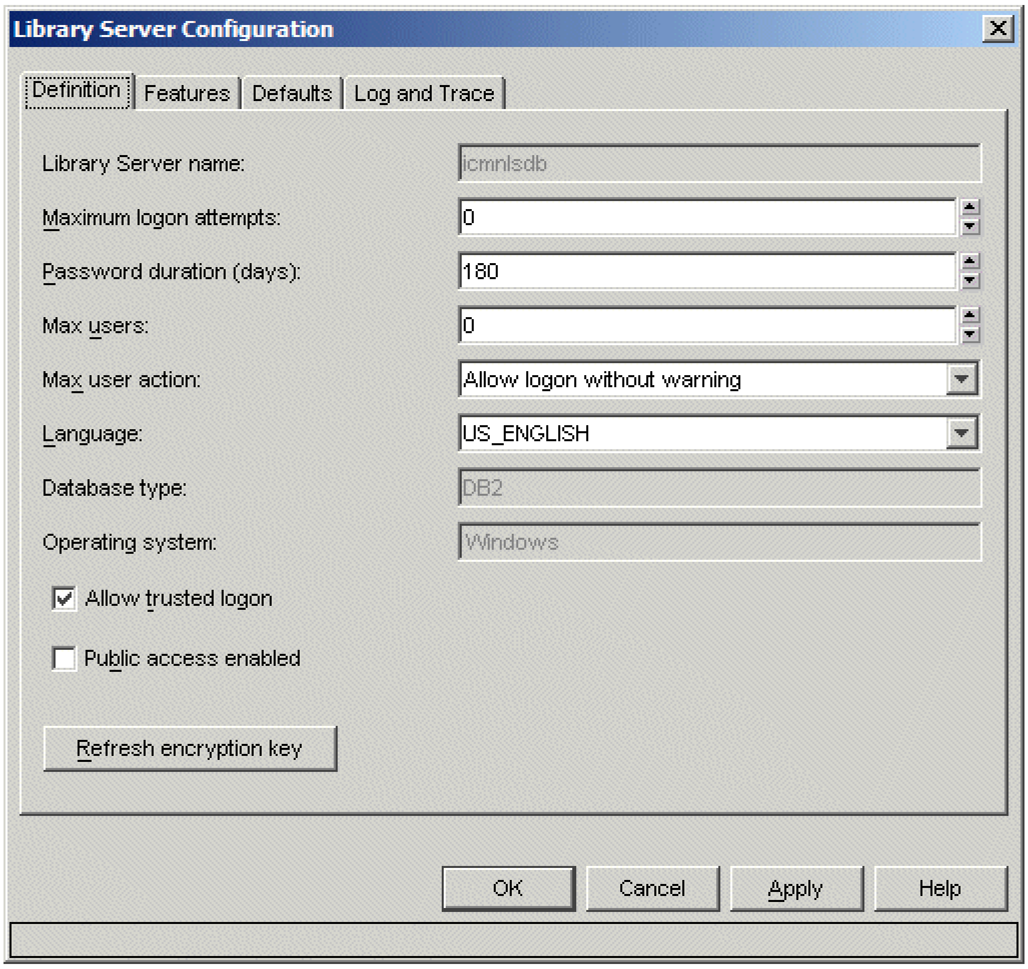
Item Type:	Adjuster Report (Content Manager V8.1 Sample Item Type)
*Adjustment Date (Content Manager V8.1 Sample Attribute) :	8/ 2/2001
*Adjuster First Name (Content Manager V8.1 Sample Attribute) :	Catherine
*Adjuster Last Name (Content Manager V8.1 Sample Attribute) :	Green
*Claimant First Name (Content Manager V8.1 Sample Attribute) :	George
*Claimant Last Name (Content Manager V8.1 Sample Attribute) :	Tolstoy
*Claim Number (Content Manager V8.1 Sample Attribute) :	4-852369
*Incident Date (Content Manager V8.1 Sample Attribute) :	7/29/2001
*License Plate Number (Content Manager V8.1 Sample Attribute) :	6USA676
*Policy Number (Content Manager V8.1 Sample Attribute) :	10285265410
Claim Amount	2,000

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

2.3 Enabling DB2 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 log on to the process environment and to the DB2 Content Manager server. This step configures DB2 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 DB2 Content Manager-based applications.

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


Library Server Configuration

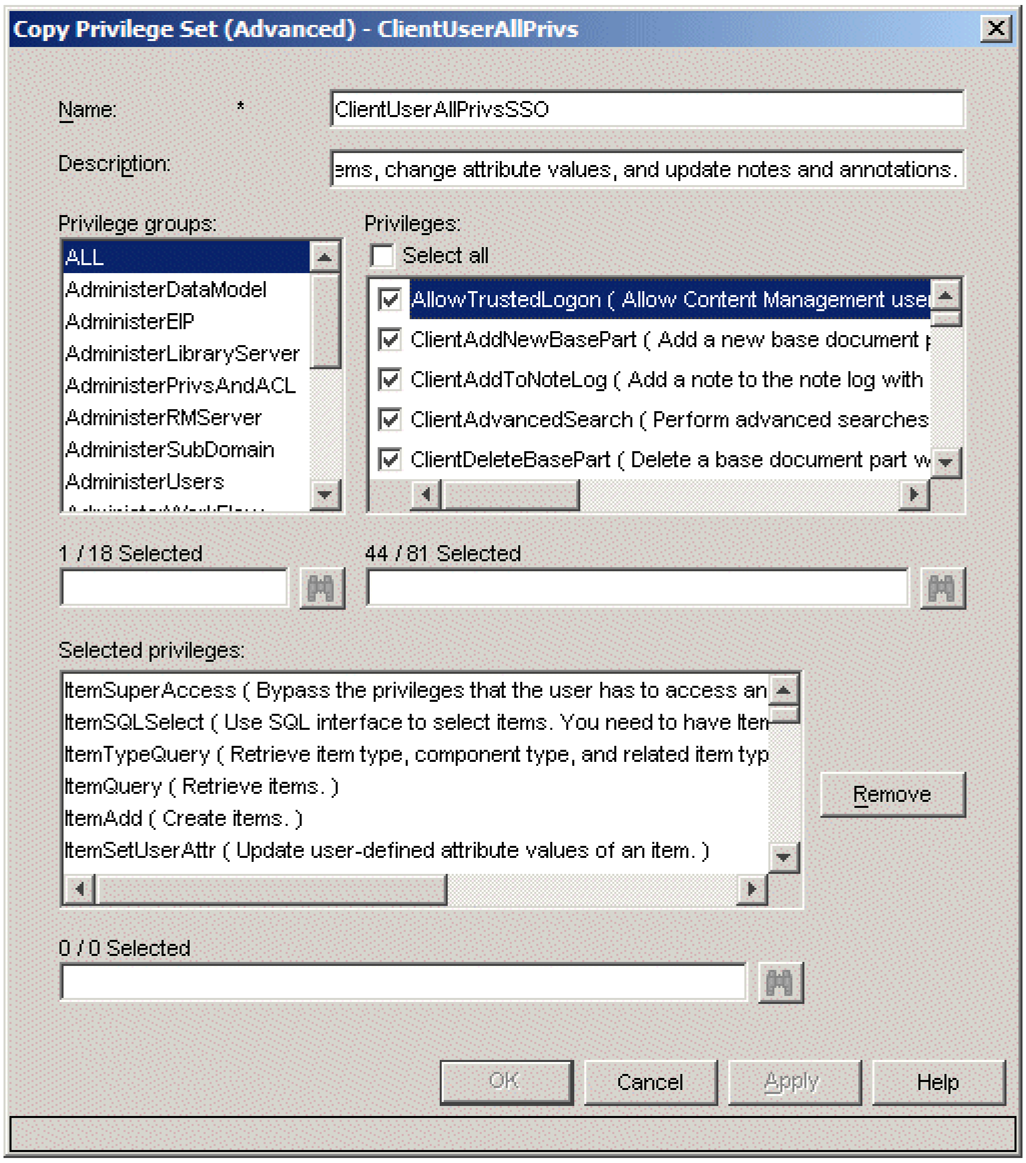
Definition | Features | Defaults | Log and Trace

Library Server name:	icmnlsdb
Maximum logon attempts:	0
Password duration (days):	180
Max users:	0
Max user action:	Allow logon without warning
Language:	US_ENGLISH
Database type:	DB2
Operating system:	Windows

☑ Allow trusted logon

☐ Public access enabled

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



Name: * ClientUserAllPrivsSSO

Description: ems, change attribute values, and update notes and annotations.

Privilege groups:

- ALL
- AdministerDataModel
- AdministerEIP
- AdministerLibraryServer
- AdministerPrivsAndACL
- AdministerRMServer
- AdministerSubDomain
- AdministerUsers

Privileges:

- [✓] AllowTrustedLogon (Allow Content Management use
- [✓] ClientAddNewBasePart (Add a new base document
- [✓] ClientAddToNoteLog (Add a note to the note log with
- [✓] ClientAdvancedSearch (Perform advanced searches
- [✓] ClientDeleteBasePart (Delete a base document part w

1 / 18 Selected

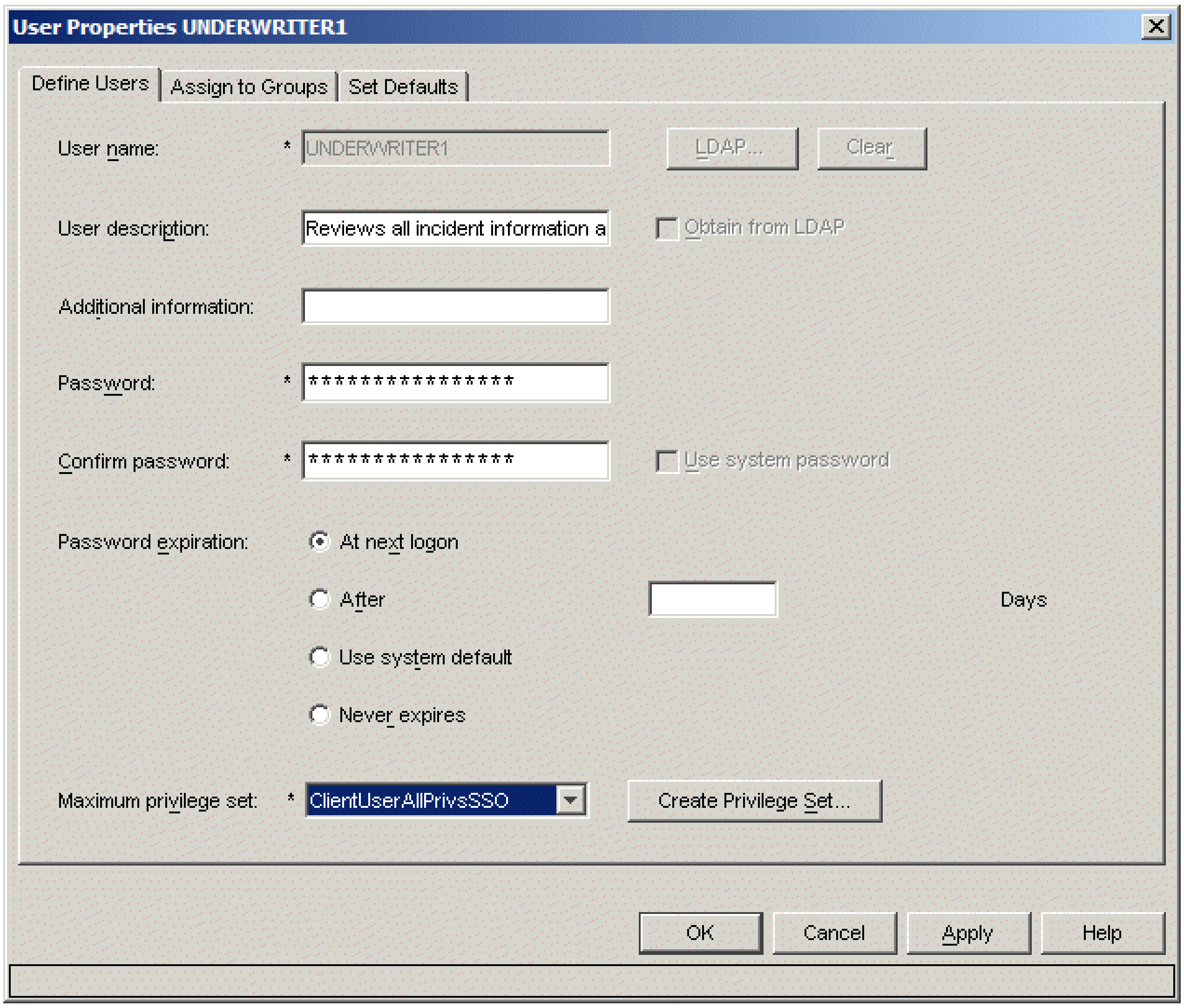
44 / 81 Selected

Selected privileges:

- ItemSuperAccess (Bypass the privileges that the user has to access an
- ItemSQLSelect (Use SQL interface to select items. You need to have Iten
- ItemTypeQuery (Retrieve item type, component type, and related item typ
- ItemQuery (Retrieve items.)
- ItemAdd (Create items.)
- ItemSetUserAttr (Update user-defined attribute values of an item.)

0 / 0 Selected

12. In the navigation panel click **Authentication > Users**. The right pane now shows the list of DB2 Content Manager users currently defined.
13. For the each of the users `Agent1`, `Agent2`, `Adjuster1`, `Adjuster2`, `Underwriter1`, `Underwriter2`, `UnderwriterAssistant1`, and `UnderwriterAssistant2` double-click the name to open the properties window and replace the value in the Maximum Privilege set field with `ClientUserAllPrivsSSO`


User Properties UNDERWRITER1

User name: * UNDERWRITER1

User description: Reviews all incident information a

[ ] Obtain from LDAP

Additional information:

Password: * ****************

Confirm password: * ****************

[ ] Use system password

Password expiration: [✓ At next logon] / After ___ Days / Use system default / Never expires

Maximum privilege set: * ClientUserAllPrivsSSO

14. Close the system administration client.

2.4 Installing supplementary files

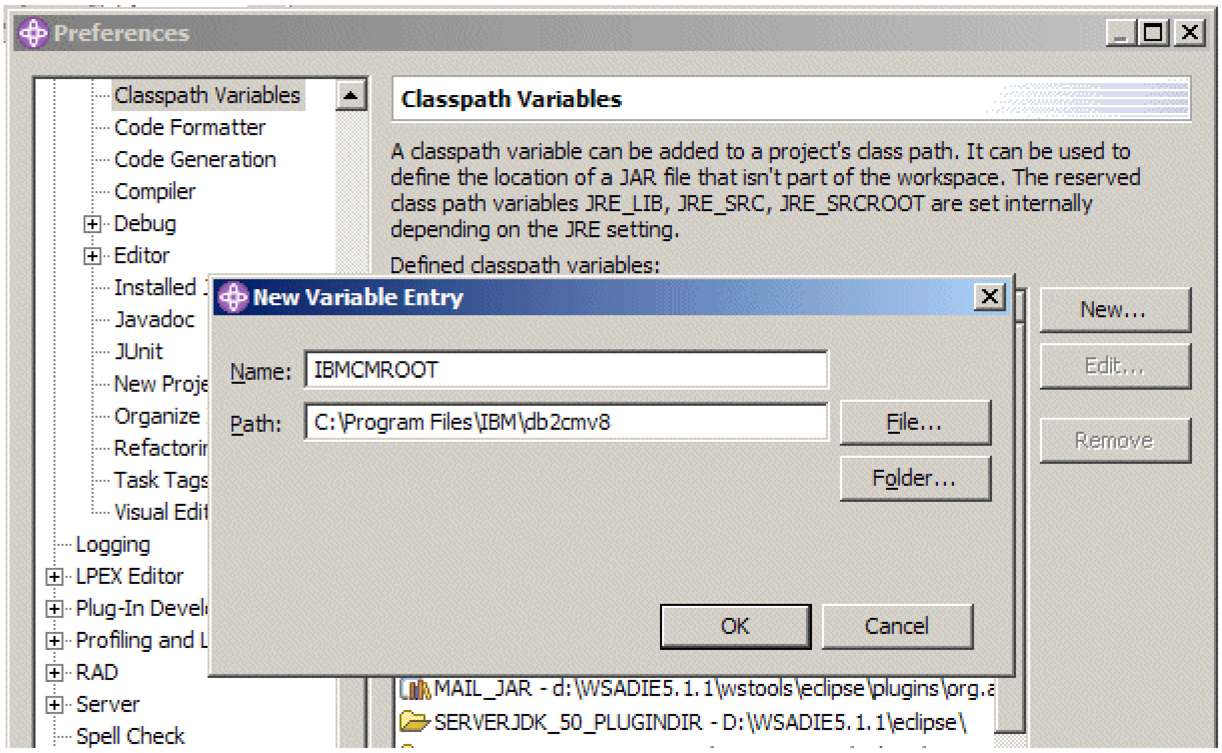
1. Unzip `quickStartSampleRev2wps.zip` into a temporary folder `<TMP>`.
2. Copy the file `bpecm.properties` from `<TMP>` to `<WPS_PROFILE_HOME>\properties`.
3. Open `bpecm.properties` with an editor and make sure the property`ContentManagerAuthenticationPwd=...` contains the password of the library server administrator (icmadmin). Note that a simple encryption method referred to as ROT13 can be used to avoid storing the password in un-encrypted form. 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.
4. Create a folder `security` under `<WPS_PROFILE_HOME>` and copy the files `users.prop` and `groups.prop` from `<TMP>` to this new directory.
5. Open `<WPS_PROFILE_HOME>\security\users.prop` in an editor, locate the line `icmadmin:passw0rd:2:2:` and replace `passw0rd` with the password of the library server administrator (icmadmin) on your system.

2.5 Configuring the Build environment

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.

1. Start *WebSphere Integration Developer* and specify a directory that serves as the `<WORKSPACE_FOLDER>`. If not prompted for a workspace, start it from the command line as follows: `<WID_HOME>\wid.exe -setworkspace <WORKSPACE_FOLDER>`.
2. On the menu select **Window > Open Perspective > Other**. The **Select Perspective** window opens. Check **Show All**, select **J2EE**, and click **OK**. Click **OK** on the **Confirm Enablement** dialogue. Close the **Welcome** view.

3. On the menu select **Window > Preferences.** The Preferences window opens.
4. Select **Java > Build Path > Classpath Variables** and click the button labeled **New**. The 'New Variable Entry' window opens.
5. Enter `IBMCMROOT` into the 'Name' field and the value of `<IBMCMROOT>` into the 'Path' field.
6. Enter `IBMWPSROOT` into the 'Name' field and the value of `<WPSROOT>` into the 'Path' field.



Classpath Variables

A classpath variable can be added to a project's class path. It can be used to define the location of a JAR file that isn't part of the workspace. The reserved class path variables JRE_LIB, JRE_SRC, JRE_SRCROOT are set internally depending on the JRE setting.

Defined classpath variables:

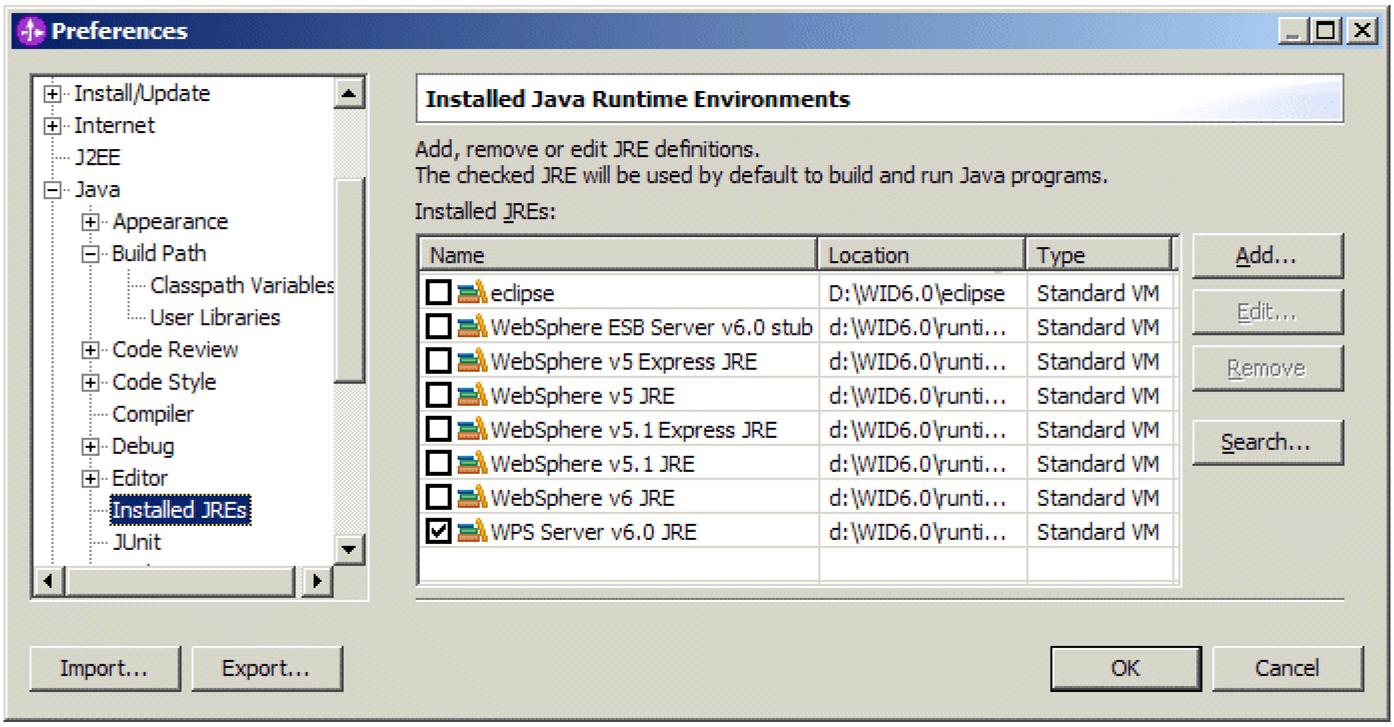
Name: IBMCMROOT

Path: C:\Program Files\IBM\db2cmv8

MAIL_JAR - d:\WSADIE5.1.1\wstools\eclipse\plugins\org.a

SERVERJDK_50_PLUGINDIR - D:\WSADIE5.1.1\eclipse\

7. Select **Java > Installed JREs** and change the JVM from default **Eclipse** to **WPS Server v6.0 JRE**



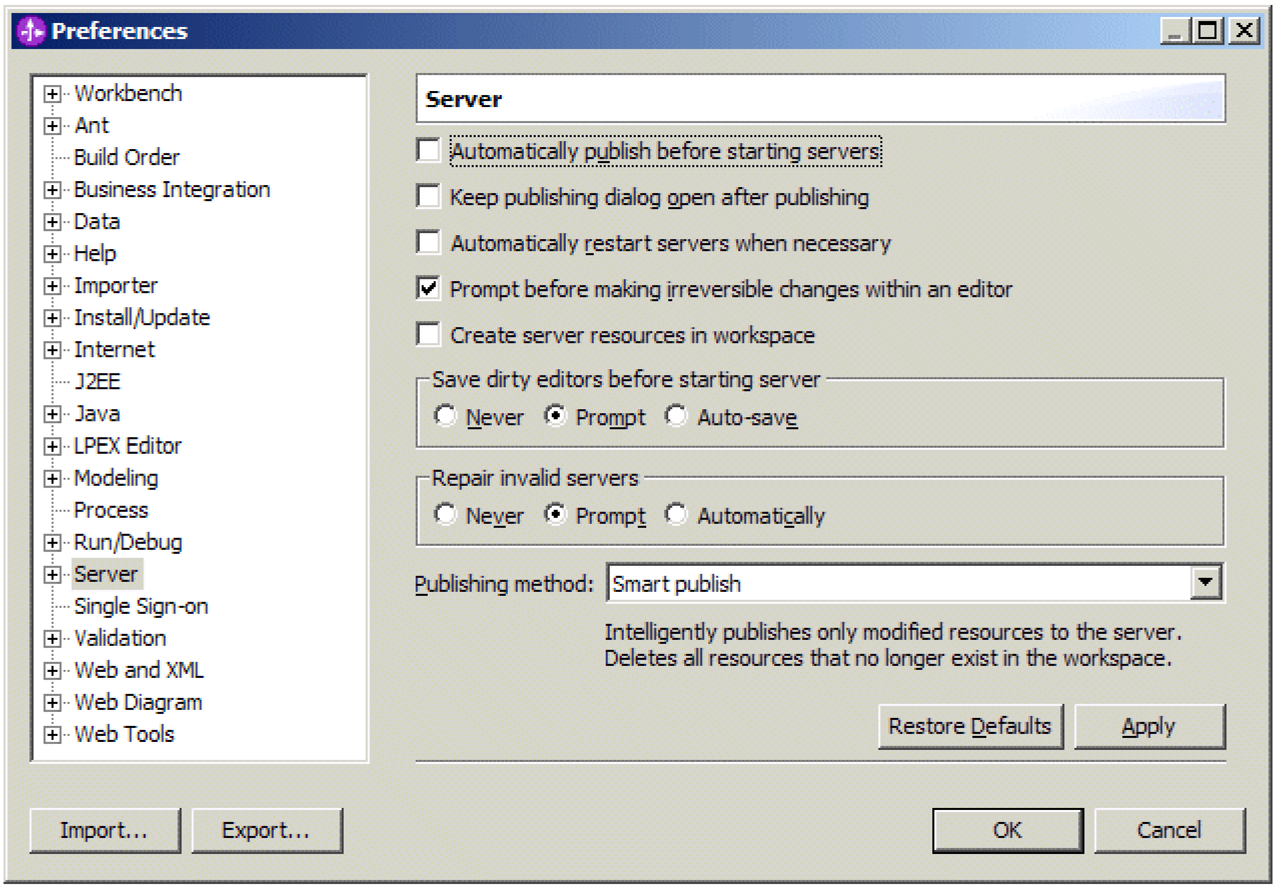
Installed Java Runtime Environments

Add, remove or edit JRE definitions.
The checked JRE will be used by default to build and run Java programs.

Installed JREs:

Name	Location	Type
☐ eclipse	D:\WID6.0\eclipse	Standard VM
☐ WebSphere ESB Server v6.0 stub	d:\WID6.0\runti...	Standard VM
☐ WebSphere v5 Express JRE	d:\WID6.0\runti...	Standard VM
☐ WebSphere v5 JRE	d:\WID6.0\runti...	Standard VM
☐ WebSphere v5.1 Express JRE	d:\WID6.0\runti...	Standard VM
☐ WebSphere v5.1 JRE	d:\WID6.0\runti...	Standard VM
☐ WebSphere v6 JRE	d:\WID6.0\runti...	Standard VM
☑ WPS Server v6.0 JRE	d:\WID6.0\runti...	Standard VM

8. Select **Server** and ensure that **Automatically publish before starting servers** is not checked.



Server

- [ ] Automatically publish before starting servers
- [ ] Keep publishing dialog open after publishing
- [ ] Automatically restart servers when necessary
- [✓] Prompt before making irreversible changes within an editor
- [ ] Create server resources in workspace

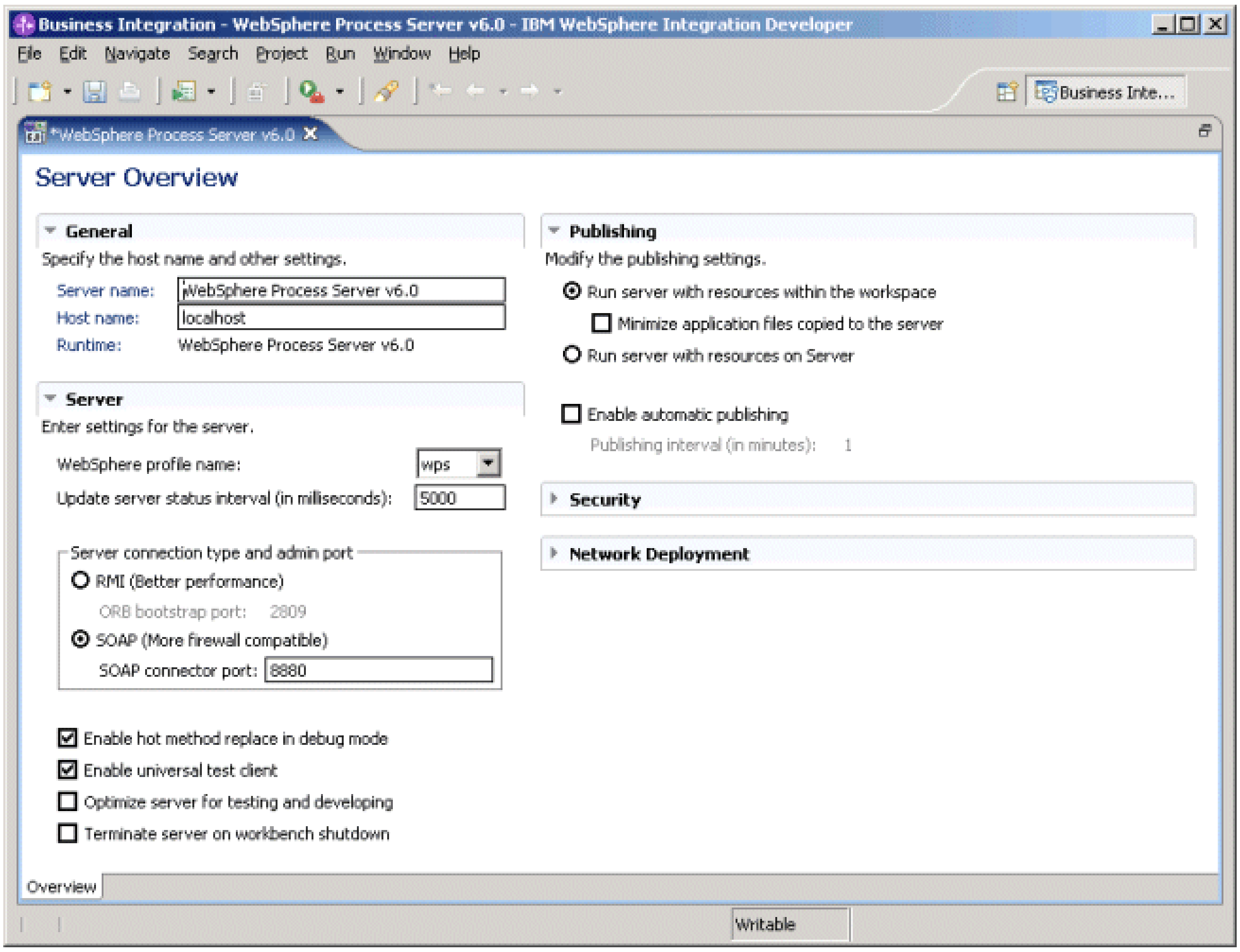
Save dirty editors before starting server: Never [✓ Prompt] Auto-save

Repair invalid servers: Never [✓ Prompt] Automatically

Publishing method: Smart publish

Intelligently publishes only modified resources to the server. Deletes all resources that no longer exist in the workspace.

9. Recommended: Select **Internet > Web Browser** and replace the default **Internal Web Browser** with a supported browser of your choice.
10. Click **OK** to close the preferences window.
11. Click the **Servers** tab to switch to the Servers view.
12. If you have installed WebSphere Integration Developer with the integrated test environment the Servers view already contains a entry for the generated profile **wps**. Double-click the entry to open the **Server Overview** editor and modify the settings as follows:
    - In the **Server connection type and admin port** section click **SOAP** as connection type and make sure the SOAP connector port matches the port settings of the Process Server profile you have created.
    - Press CTRL-S to save your changes and close the editor.
13. If you have installed WebSphere Process Server separately, you need to create an entry in the server view as follows:
    - Right-click the blank area of the server view and select **New > Server**. On the **New Server** window select **WebSphere Process v6.0 Server** from the list of server types. The default host name `localhost` should be OK.
    - Click **Next** and specify or verify the path to the install directory. It should point to `<WPS_HOME>`.
    - Click **Next**. On the **WebSphere Server Settings** page select **SOAP** as connection type. And verify that the **WebSphere profile name** points to the profile you have created.
    - Click **Finish** to complete the server definition.


Business Integration - WebSphere Process Server v6.0 - IBM WebSphere Integration Developer

File Edit Navigate Search Project Run Window Help

*WebSphere Process Server v6.0

Server Overview

General
Specify the host name and other settings.

Server name:	WebSphere Process Server v6.0
Host name:	localhost
Runtime:	WebSphere Process Server v6.0

Server
Enter settings for the server.

WebSphere profile name:	wps
Update server status interval (in milliseconds):	5000

Server connection type and admin port
- RMI (Better performance)
  - ORB bootstrap port: 2809
- [✓ SOAP (More firewall compatible)]
  - SOAP connector port: 8880

- [✓ Enable hot method replace in debug mode]
- [✓ Enable universal test client]
- Optimize server for testing and developing
- Terminate server on workbench shutdown

Publishing
Modify the publishing settings.
- [✓ Run server with resources within the workspace]
  - Minimize application files copied to the server
- Run server with resources on Server

- Enable automatic publishing
  - Publishing interval (in minutes): 1

Security

Network Deployment

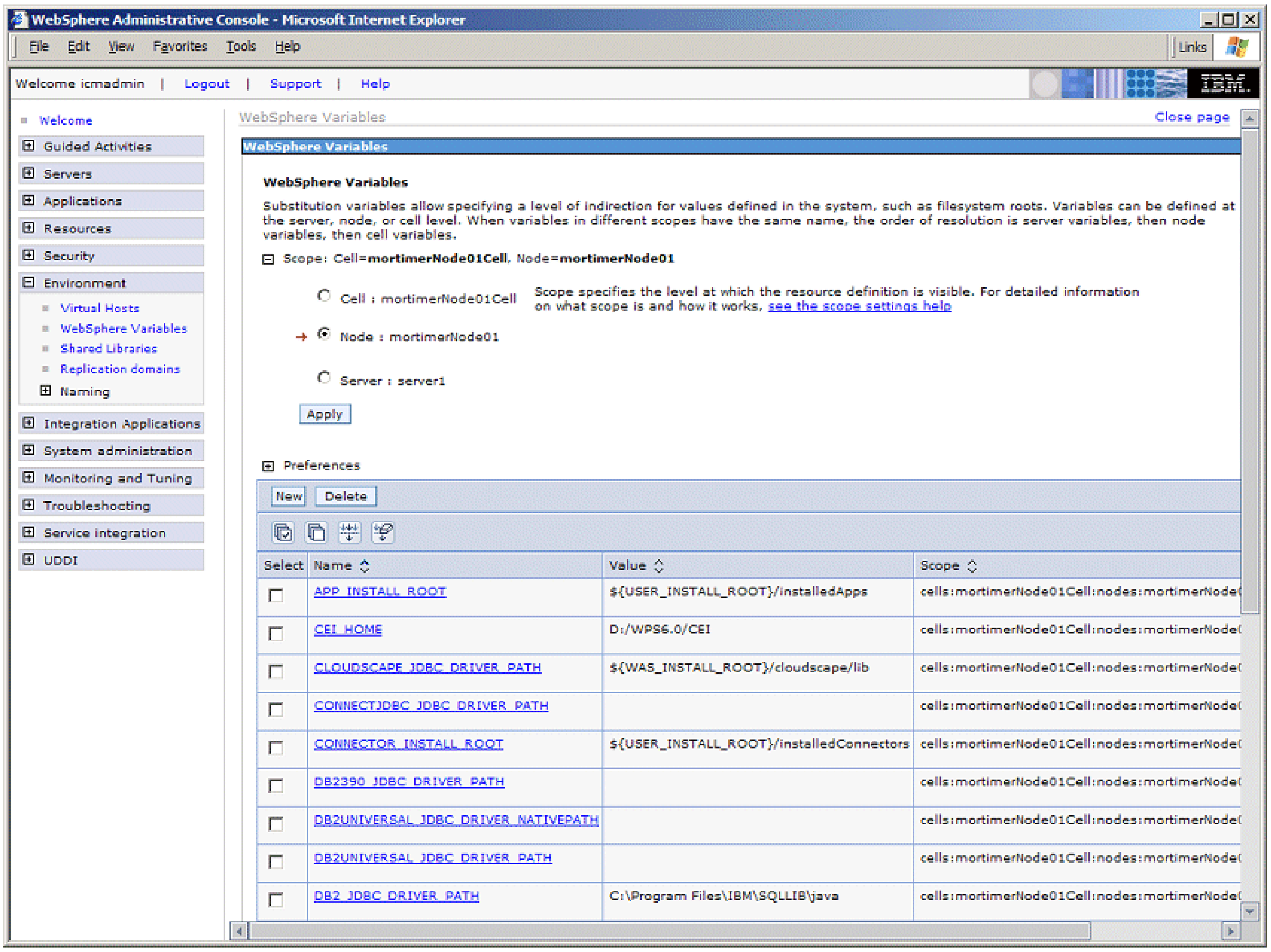
Overview

Writable

2.6 Setting up the test environment

The Integration uses a JDBC connection to access the library server database. This connection and other settings such as global security need to be configured with the Process Server's administrative console.

1. Perform the steps in 6.2 Starting the Process Server and launching the administrative console and log on as `icmadmin`. Since security is not yet enabled this only serves as a name for your session. In case you close the admin console without saving your changes you will be able to recover by logging on with this same userid again.
2. In the navigation panel select `Environment > WebSphere Variables` and set the scope to `Node` level by checking the radio button labeled **Node** and clicking **Apply**.
3. Ensure that the variable `DB2_JDBC_DRIVER_PATH` is present (you may need to switch the scope to **Node**) and that it points to `<DB2HOME>/java`



WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Scope: Cell=**mortimerNode01Cell**, Node=**mortimerNode01**

( ) Cell : mortimerNode01Cell
(•) Node : mortimerNode01
( ) Server : server1

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help

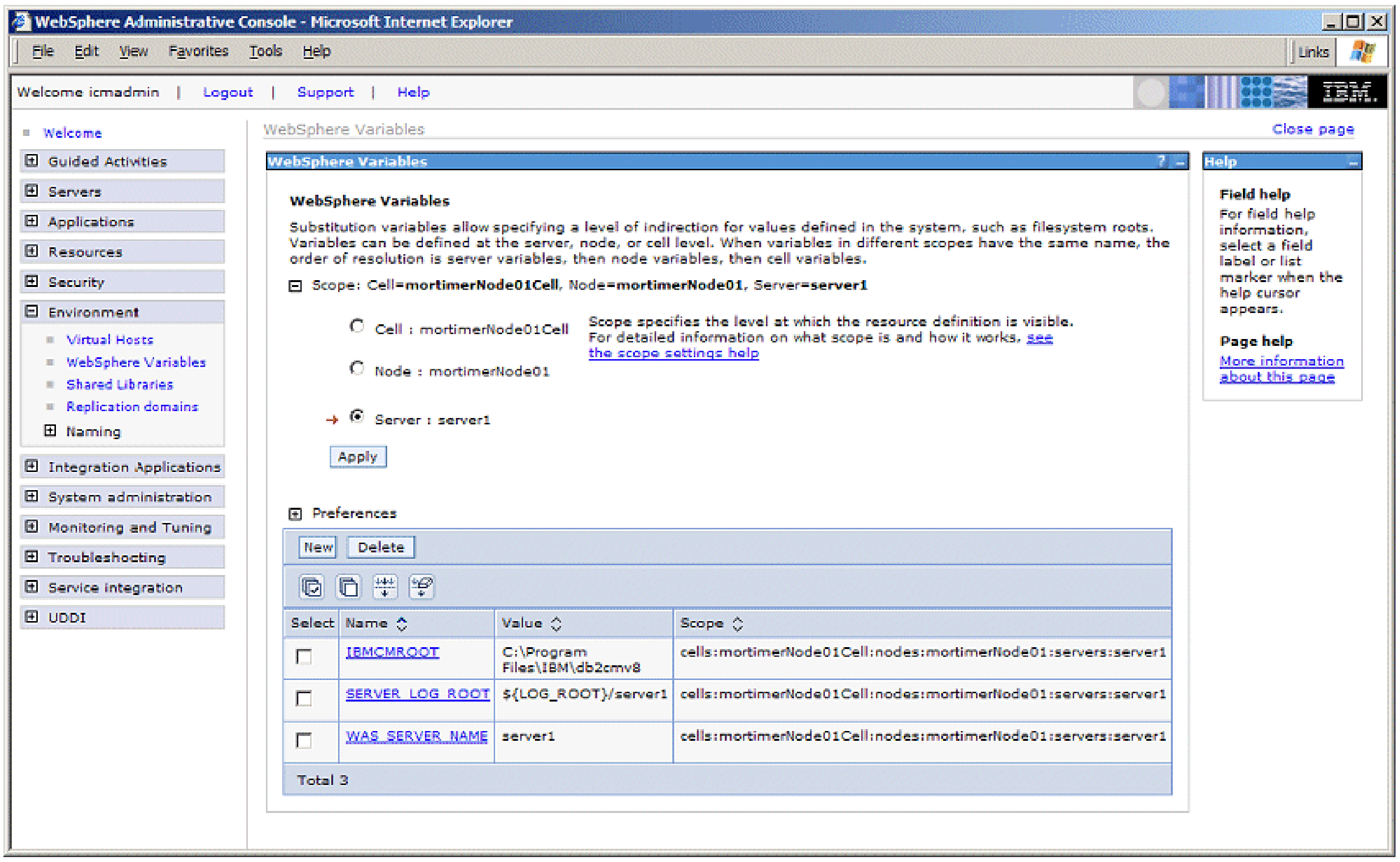
Apply

Preferences

New Delete

Select	Name	Value	Scope
☐	APP_INSTALL_ROOT	${USER_INSTALL_ROOT}/installedApps	cells:mortimerNode01Cell:nodes:mortimerNode
☐	CEI_HOME	D:/WPS6.0/CEI	cells:mortimerNode01Cell:nodes:mortimerNode
☐	CLOUDSCAPE_JDBC_DRIVER_PATH	${WAS_INSTALL_ROOT}/cloudscape/lib	cells:mortimerNode01Cell:nodes:mortimerNode
☐	CONNECTJDBC_JDBC_DRIVER_PATH		cells:mortimerNode01Cell:nodes:mortimerNode
☐	CONNECTOR_INSTALL_ROOT	${USER_INSTALL_ROOT}/installedConnectors	cells:mortimerNode01Cell:nodes:mortimerNode
☐	DB2390_JDBC_DRIVER_PATH		cells:mortimerNode01Cell:nodes:mortimerNode
☐	DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH		cells:mortimerNode01Cell:nodes:mortimerNode
☐	DB2UNIVERSAL_JDBC_DRIVER_PATH		cells:mortimerNode01Cell:nodes:mortimerNode
☐	DB2_JDBC_DRIVER_PATH	C:\Program Files\IBM\SQLLIB\java	cells:mortimerNode01Cell:nodes:mortimerNode

4. Create a new variable `IBMCMROOT`. In the **Scope** section click **Server: server1** and **Apply**. Click **New**, enter `IBMCMROOT` into the **Name** field, and `<IBMCMROOT>` into the **Value** field. Click **OK** to store the new variable.



WebSphere Variables

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Scope: Cell=**mortimerNode01Cell**, Node=**mortimerNode01**, Server=**server1**

( ) Cell : mortimerNode01Cell
( ) Node : mortimerNode01
(•) Server : server1

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help

Apply

Preferences

New Delete

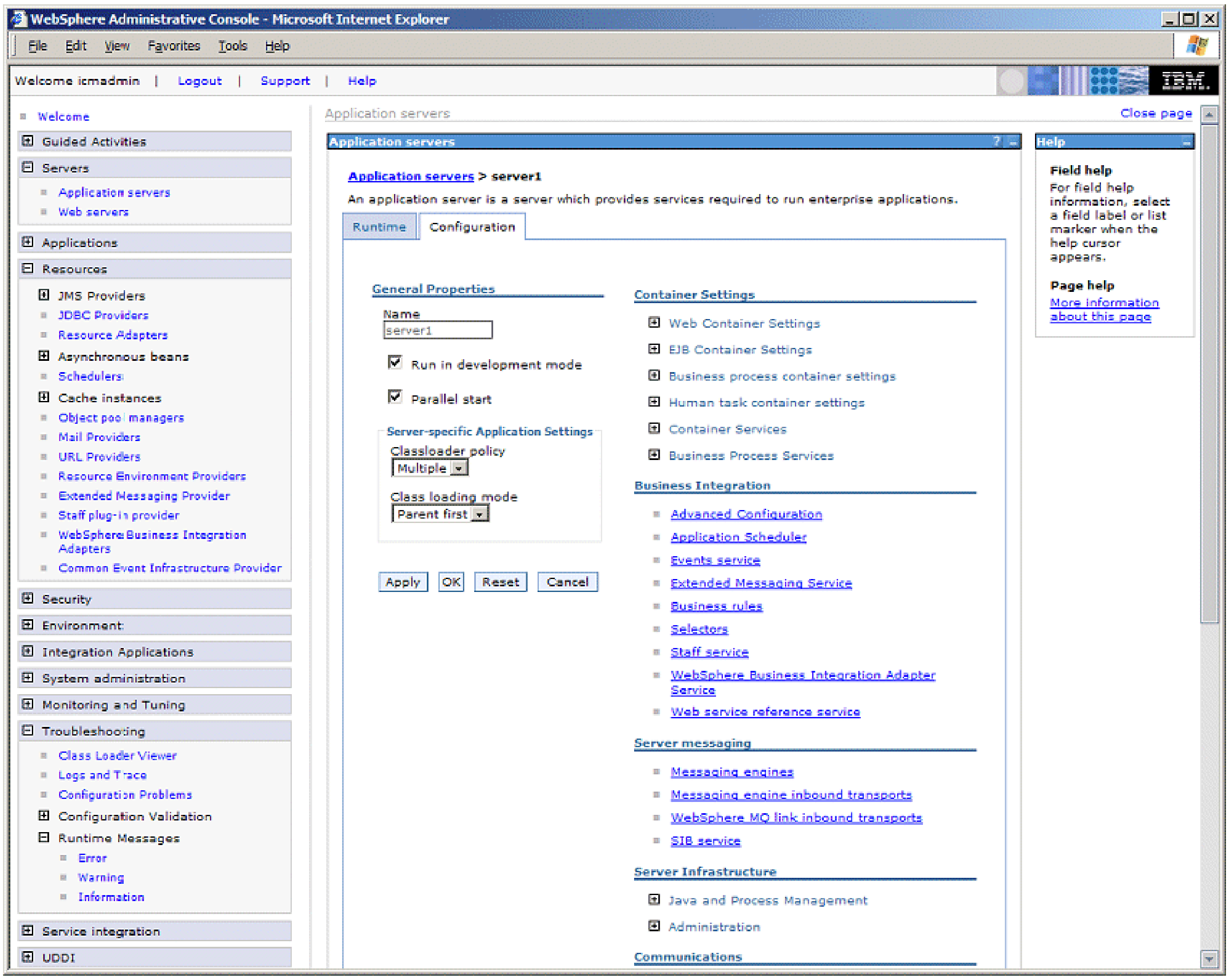
Select	Name	Value	Scope
☐	IBMCMROOT	C:\Program Files\IBM\db2cmv8	cells:mortimerNode01Cell:nodes:mortimerNode01:servers:server1
☐	SERVER_LOG_ROOT	${LOG_ROOT}/server1	cells:mortimerNode01Cell:nodes:mortimerNode01:servers:server1
☐	WAS_SERVER_NAME	server1	cells:mortimerNode01Cell:nodes:mortimerNode01:servers:server1

Total 3

Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
More information about this page

5. In the navigation panel select `Servers > Application Servers`, click **server1**.
6. If **Run in development mode** is not checked, check it and click **Apply**. It is checked by default for the integrated test environment but it must be explicitly enabled when using a separate stand-alone installation of WebSphere Process Server. Setting it simplifies development and re-deployment of changed modules as process data does not need to be explicitly deleted. Note that in development mode changes to a deployed process automatically trigger the deletion of any existing process or activity instances. If development mode is not enabled, process instances need to be deleted (e.g. using the BPC Explorer) before the process template can be stopped. The process template needs to be stopped before the process can be un-deployed if development mode is not enabled. See section 6.7 on starting and stopping the process application manually.



Application servers

Application servers > server1

An application server is a server which provides services required to run enterprise applications.

General Properties

Name: server1

☑ Run in development mode

☑ Parallel start

Server-specific Application Settings

Classloader policy: Multiple

Class loading mode: Parent first

Container Settings
- Web Container Settings
- EJB Container Settings
- Business process container settings
- Human task container settings
- Container Services
- Business Process Services

Business Integration
- Advanced Configuration
- Application Scheduler
- Events service
- Extended Messaging Service
- Business rules
- Selectors
- Staff service
- WebSphere Business Integration Adapter Service
- Web service reference service

Server messaging
- Messaging engines
- Messaging engine inbound transports
- WebSphere MQ link inbound transports
- SIB service

Server Infrastructure
- Java and Process Management
- Administration

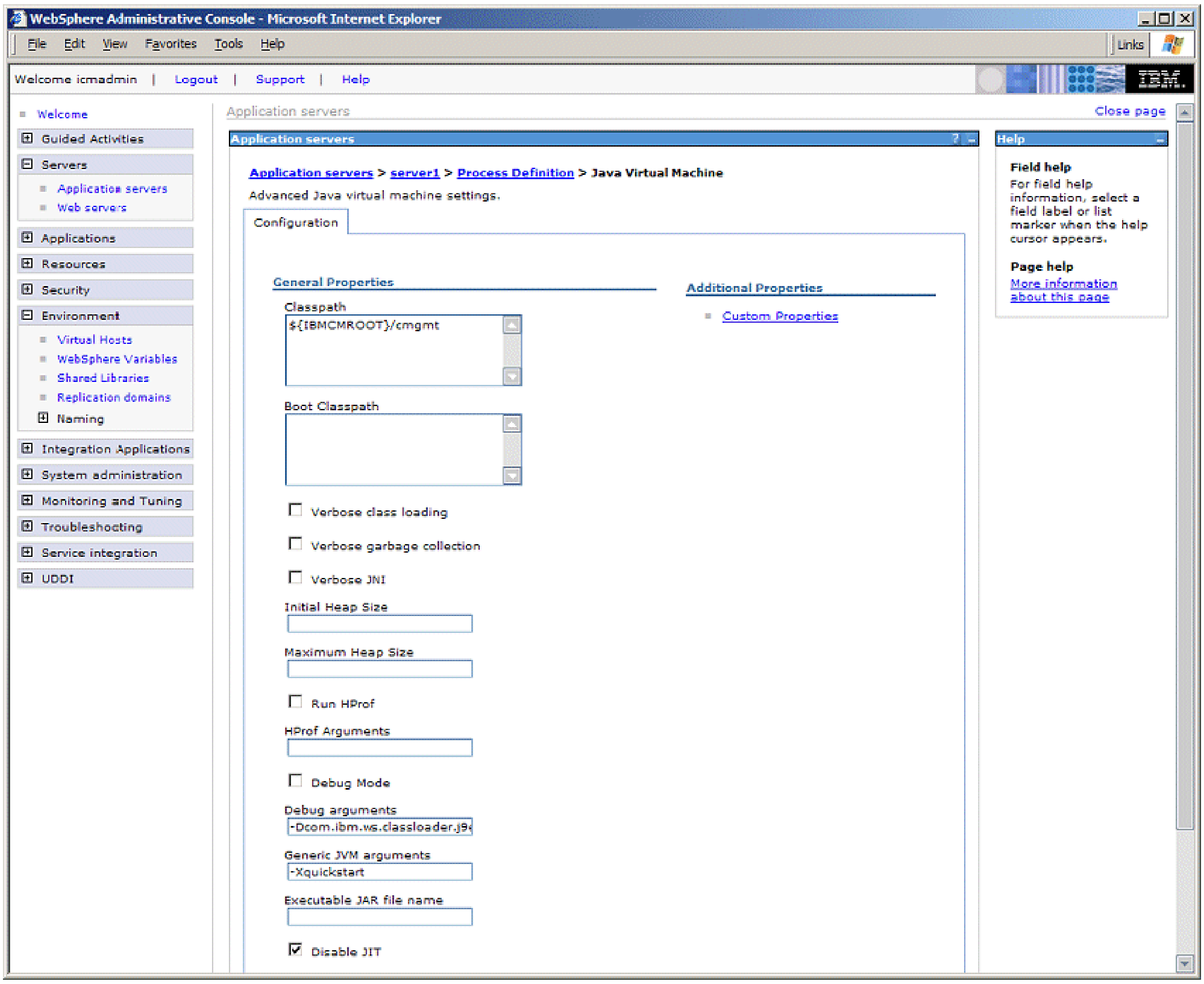
Communications

Help

Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
More information about this page

7. On the configuration page locate the section **Server Infrastructure**, expand **Java and Process Management**, and select **Process Definition**. In the **Additional Properties** section of the **Process Definition** page select **Java Virtual Machine**.
8. If you have not updated the WAS JRE as described in 6.4 you need to check **Disable JIT** to avoid a problem that might cause the server to shut down. However, we strongly recommend to update the WAS JRE as described in 6.4 since disabling the JIT has a significant performance impact.
9. Enter `${IBMCMROOT}/cmgmt` into the **Classpath** field and click **Apply**


WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Welcome icmadmin | Logout | Support | Help

Welcome
Guided Activities
Servers
- Application servers
- Web servers

Applications
Resources
Security
Environment
- Virtual Hosts
- WebSphere Variables
- Shared Libraries
- Replication domains
- Naming

Integration Applications
System administration
Monitoring and Tuning
Troubleshooting
Service integration
UDDI

Application servers | Close page

Application servers

Application servers > server1 > Process Definition > Java Virtual Machine

Advanced Java virtual machine settings.

Configuration

General Properties

Classpath: ${IBMCMROOT}/cmgmt

Boot Classpath:

☐ Verbose class loading

☐ Verbose garbage collection

☐ Verbose JNI

Initial Heap Size:

Maximum Heap Size:

☐ Run HProf

HProf Arguments:

☐ Debug Mode

Debug arguments: -Dcom.ibm.ws.classloader.j9

Generic JVM arguments: -Xquickstart

Executable JAR file name:

☑ Disable JIT

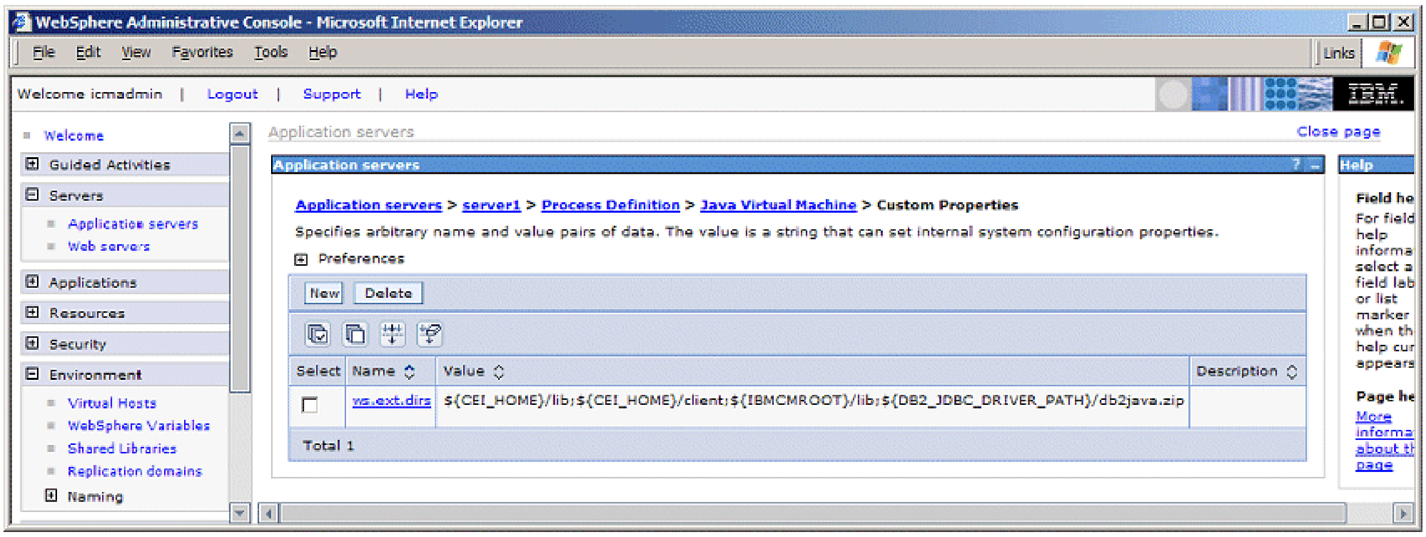
Additional Properties
- Custom Properties

Help

Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
More information about this page

10. In the **Additional Properties** section select **Custom Properties**.
11. If a property with the name `ws.ext.dirs` exists, add a semicolon followed by `${IBMCMROOT}/lib;${DB2_JDBC_DRIVER_PATH}/db2java.zip` to the end of its value. If no such property exists, create a new one with this value.


WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Welcome icmadmin | Logout | Support | Help

Application servers | Close page

Application servers

Application servers > server1 > Process Definition > Java Virtual Machine > Custom Properties

Specifies arbitrary name and value pairs of data. The value is a string that can set internal system configuration properties.

Preferences

New | Delete

Select	Name	Value	Description
☐	ws.ext.dirs	${CEI_HOME}/lib;${CEI_HOME}/client;${IBMCMROOT}/lib;${DB2_JDBC_DRIVER_PATH}/db2java.zip	
Total 1			

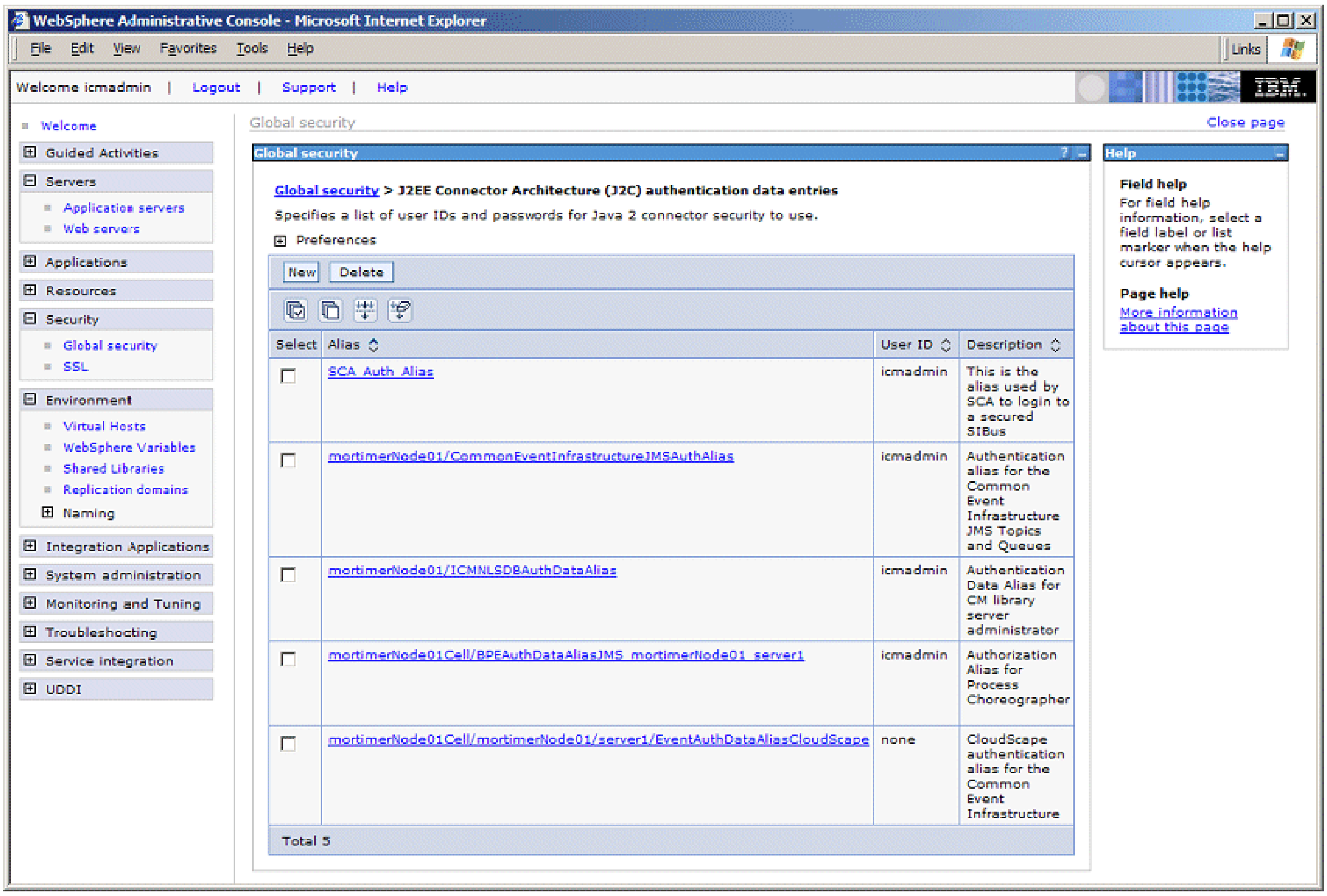
12. Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.

2.7 Configure JDBC connections and data sources

1. In the navigation panel select **Security > Global Security**.

2. In the **Authentication** section select **JAAS Configuration > J2C Authentication data**.
3. Verify the existence of the authentication alias `<HOSTNAME><NODENAME><CELLNAME>/BPEAuthDataAliasJMS_<HOSTNAME>_server1` that has been created during process container configuration.
4. Click **New**, add the following entries and click **OK** to store them.

Alias	`ICMNLSDBAuthDataAlias`
User ID	`icmadmin`
Password	icmadmin's password
Description (optional)	Authentication data alias for CM library server administrator



Global security > J2EE Connector Architecture (J2C) authentication data entries

Specifies a list of user IDs and passwords for Java 2 connector security to use.

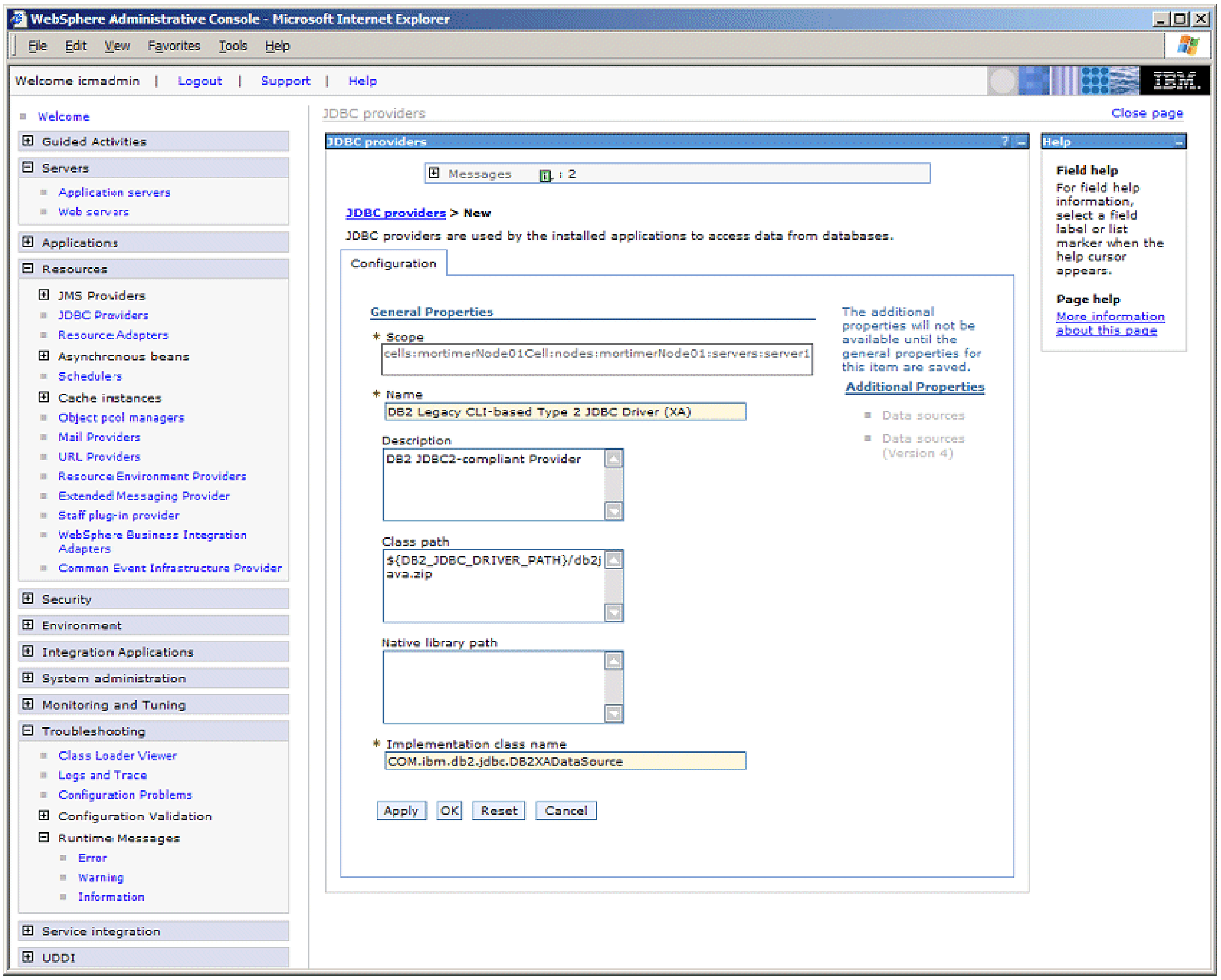
Select	Alias	User ID	Description
☐	SCA_Auth_Alias	icmadmin	This is the alias used by SCA to login to a secured SIBus
☐	mortimerNode01/CommonEventInfrastructureJMSAuthAlias	icmadmin	Authentication alias for the Common Event Infrastructure JMS Topics and Queues
☐	mortimerNode01/ICMNLSDBAuthDataAlias	icmadmin	Authentication Data Alias for CM library server administrator
☐	mortimerNode01Cell/BPEAuthDataAliasJMS_mortimerNode01_server1	icmadmin	Authorization Alias for Process Choreographer
☐	mortimerNode01Cell/mortimerNode01/server1/EventAuthDataAliasCloudScape	none	CloudScape authentication alias for the Common Event Infrastructure

Total 5

Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
More information about this page

5. In the navigation panel select **Resources > JDBC Providers**. Ensure the scope is on `server1`.
6. If `DB2 Legacy CLI-based Type 2 JDBC Driver (XA)` is not listed on the cell, node, or server1 level, perform the following steps to add it at the server1 level:
    1. Set the scope to server1 and click **OK**.
    2. Step1: select the database type **DB2**
    3. Step2: select the provider type **DB2 Legacy CLI-based Type 2 JDBC Driver**
    4. Step3: select **XA data source**
    5. Click **Next**
    6. Verify that the **Class path** field contains `${DB2_JDBC_DRIVER_PATH}/db2java.zip` and that the **Implementation class name** is `COM.ibm.db2.jdbc.DB2XADataSource`.
    7. Click **OK** to add the JDBC provider to the list of providers available at server scope.



JDBC providers > New

JDBC providers are used by the installed applications to access data from databases.

General Properties

* Scope
cells:mortimerNode01Cell:nodes:mortimerNode01:servers:server1

* Name
DB2 Legacy CLI-based Type 2 JDBC Driver (XA)

Description
DB2 JDBC2-compliant Provider

Class path
${DB2_JDBC_DRIVER_PATH}/db2java.zip

Native library path

* Implementation class name
COM.ibm.db2.jdbc.DB2XADataSource

The additional properties will not be available until the general properties for this item are saved.

Additional Properties
- Data sources
- Data sources (Version 4)

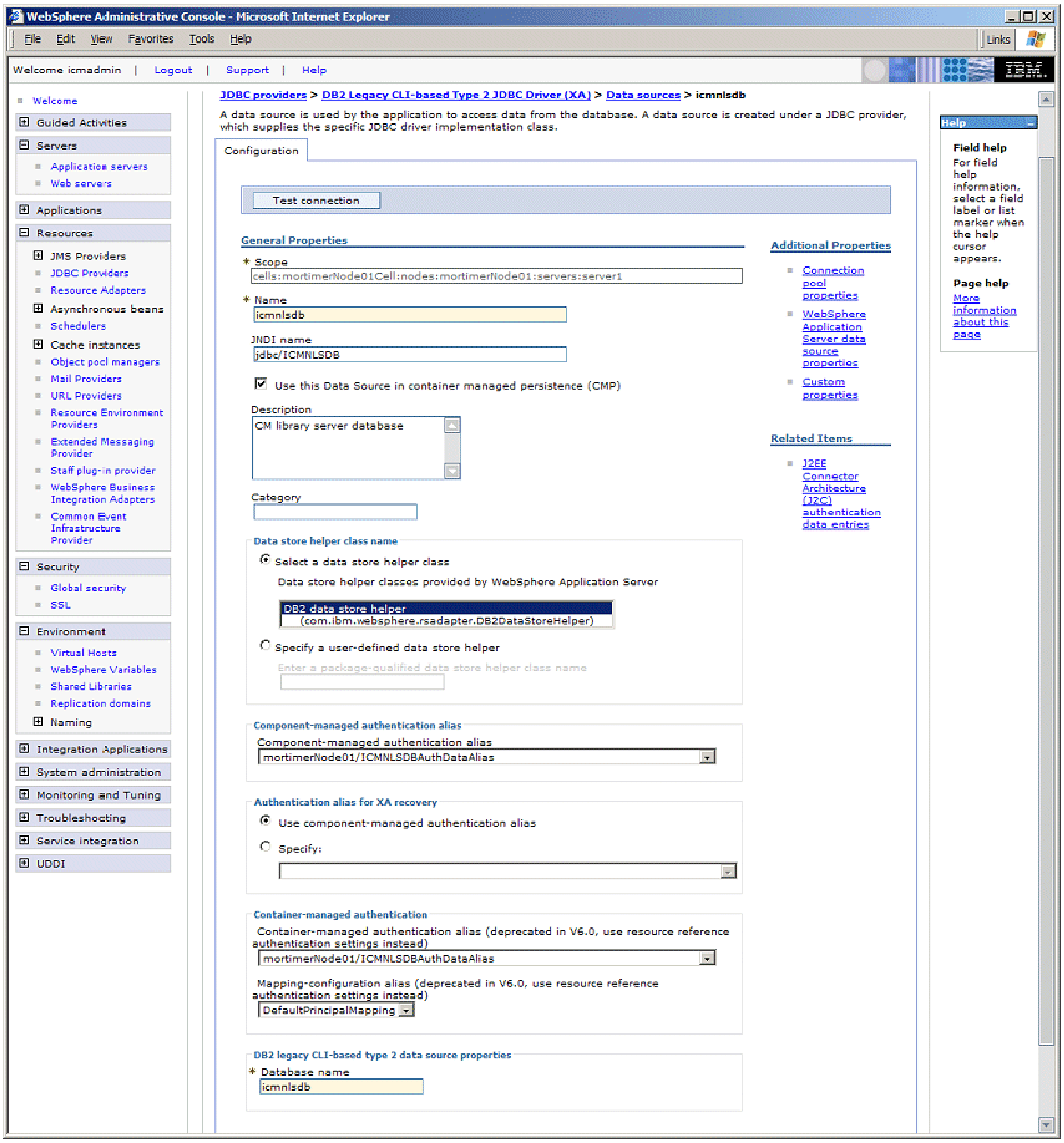
Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
More information about this page

7. Click `DB2 Legacy CLI-based Type 2 JDBC Driver (XA)` and click **Data Sources** in the **Additional Properties** to display the data sources that are based on this JDBC provider. If the JDBC provider has just been added, the list will be empty.
8. Click **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 `icmnlsdb`
JNDI Name	replace `jdbc/DB2 Legacy CLI-based JDBC Driver XA DataSource` with `jdbc/ICMNLSDB`
Description (optional)	replace `New JDBC Datasource` with `CM library server`
Component-managed Authentication Alias	select `<HOSTNAME><NODE>/ICMNLSDBAuthDataAlias` from the list
Container-managed Authentication Alias	select `<HOSTNAME><NODE>/ICMNLSDBAuthDataAlias` from the list
Mapping-configuration alias	select `DefaultPrincipalMapping` from the list
Database name	icmnlsdb

9. Click **OK** to save the data source definition.


WebSphere Administrative Console - Microsoft Internet Explorer

Welcome icmadmin | Logout | Support | Help

JDBC providers > DB2 Legacy CLI-based Type 2 JDBC Driver (XA) > Data sources > icmnlsdb

A data source is used by the application to access data from the database. A data source is created under a JDBC provider, which supplies the specific JDBC driver implementation class.

Configuration

Test connection

General Properties

* Scope: cells:mortimerNode01Cell:nodes:mortimerNode01:servers:server1

* Name: icmnlsdb

JNDI name: jdbc/ICMNLSDB

[✓] Use this Data Source in container managed persistence (CMP)

Description: CM library server database

Category:

Data store helper class name

(●) Select a data store helper class

Data store helper classes provided by WebSphere Application Server

DB2 data store helper (com.ibm.websphere.rsadapter.DB2DataStoreHelper)

( ) Specify a user-defined data store helper

Enter a package-qualified data store helper class name

Component-managed authentication alias

Component-managed authentication alias: mortimerNode01/ICMNLSDBAuthDataAlias

Authentication alias for XA recovery

(●) Use component-managed authentication alias

( ) Specify:

Container-managed authentication

Container-managed authentication alias (deprecated in V6.0, use resource reference authentication settings instead): mortimerNode01/ICMNLSDBAuthDataAlias

Mapping-configuration alias (deprecated in V6.0, use resource reference authentication settings instead): DefaultPrincipalMapping

DB2 legacy CLI-based type 2 data source properties

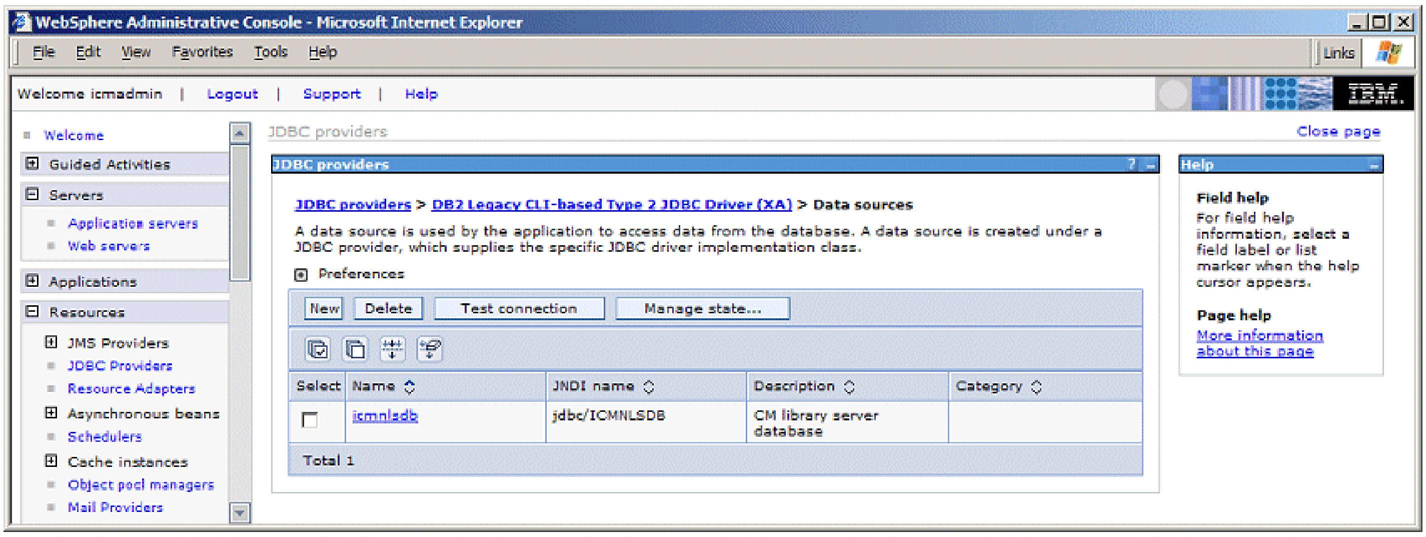
* Database name: icmnlsdb

Additional Properties: Connection pool properties; WebSphere Application Server data source properties; Custom properties

Related Items: J2EE Connector Architecture (J2C) authentication data entries

Help — Field help: For field help information, select a field label or list marker when the help cursor appears. Page help: More information about this page

10. Back on the **Data Sources** page click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.
11. Perform the following steps to verify if the data source can be accessed:
    1. Select the check box left of `icmnlsdb`.
    2. Click **Test Connection**.
    3. Verify that the information message on top of the page says `Test Connection for datasource icmnlsdb on server server1 at node <HOSTNAME><NODE> was successful`.



JDBC providers > DB2 Legacy CLI-based Type 2 JDBC Driver (XA) > Data sources

A data source is used by the application to access data from the database. A data source is created under a JDBC provider, which supplies the specific JDBC driver implementation class.

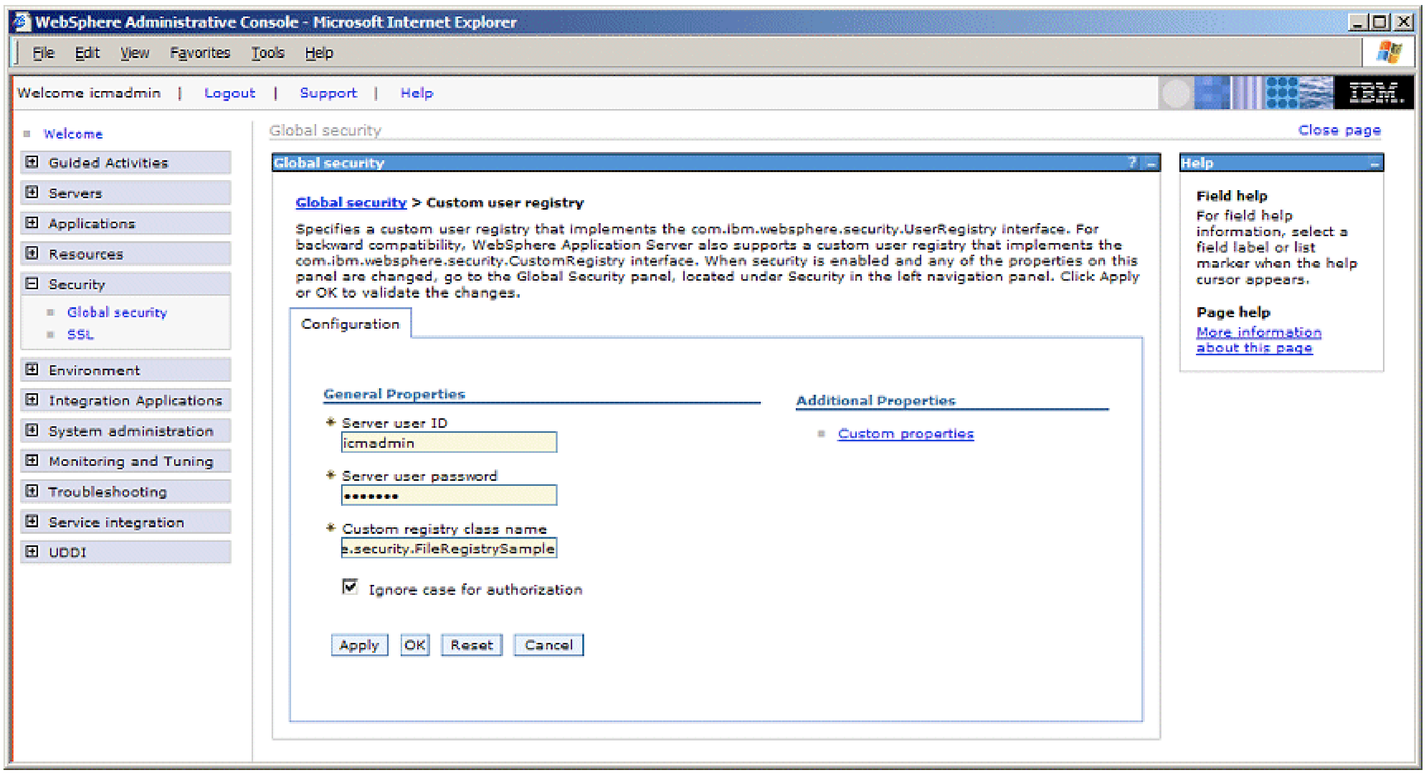
Select	Name	JNDI name	Description	Category
☐	icmnlsdb	jdbc/ICMNLSDB	CM library server database	

Total 1

2.8 Configure security

The following steps need to be performed for both the integrated test environment and a separately installed WebSphere Process Server.

1. In the navigation panel select **Security > Global Security**.
2. In the **User Registries** section to the right select **Custom**.
3. Enter `icmadmin` into the **Server user ID** field and the corresponding password into the **Server user password** field.
4. Enter `com.ibm.websphere.security.FileRegistrySample` into the **Custom registry class name** field.
5. Check **Ignore case for authorization**.
6. Click **Apply**.



Global security > Custom user registry

Specifies a custom user registry that implements the com.ibm.websphere.security.UserRegistry interface. For backward compatibility, WebSphere Application Server also supports a custom user registry that implements the com.ibm.websphere.security.CustomRegistry interface. When security is enabled and any of the properties on this panel are changed, go to the Global Security panel, located under Security in the left navigation panel. Click Apply or OK to validate the changes.

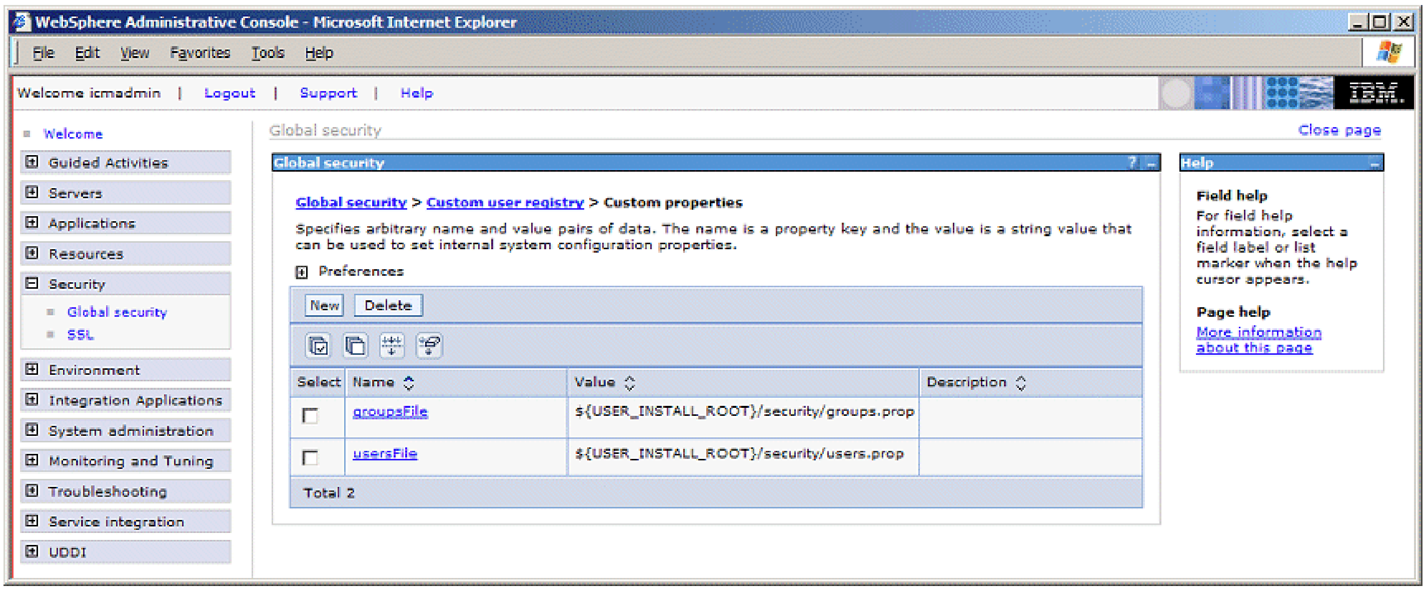
* Server user ID: icmadmin

* Server user password: •••••••

* Custom registry class name: e.security.FileRegistrySample

☑ Ignore case for authorization

7. In the **Additional Properties** section click **Custom Properties**.
8. Create a new property with name `groupsFile` and value `${USER_INSTALL_ROOT}/security/groups.prop`.
9. Create a new property with name `usersFile` and value `${USER_INSTALL_ROOT}/security/users.prop`.
10. Click **OK**.



WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Welcome icmadmin | Logout | Support | Help

Welcome
Guided Activities
Servers
Applications
Resources
Security
Global security
SSL
Environment
Integration Applications
System administration
Monitoring and Tuning
Troubleshooting
Service integration
UDDI

Global security | Close page

Global security

Global security > Custom user registry > **Custom properties**

Specifies arbitrary name and value pairs of data. The name is a property key and the value is a string value that can be used to set internal system configuration properties.

Preferences

New | Delete

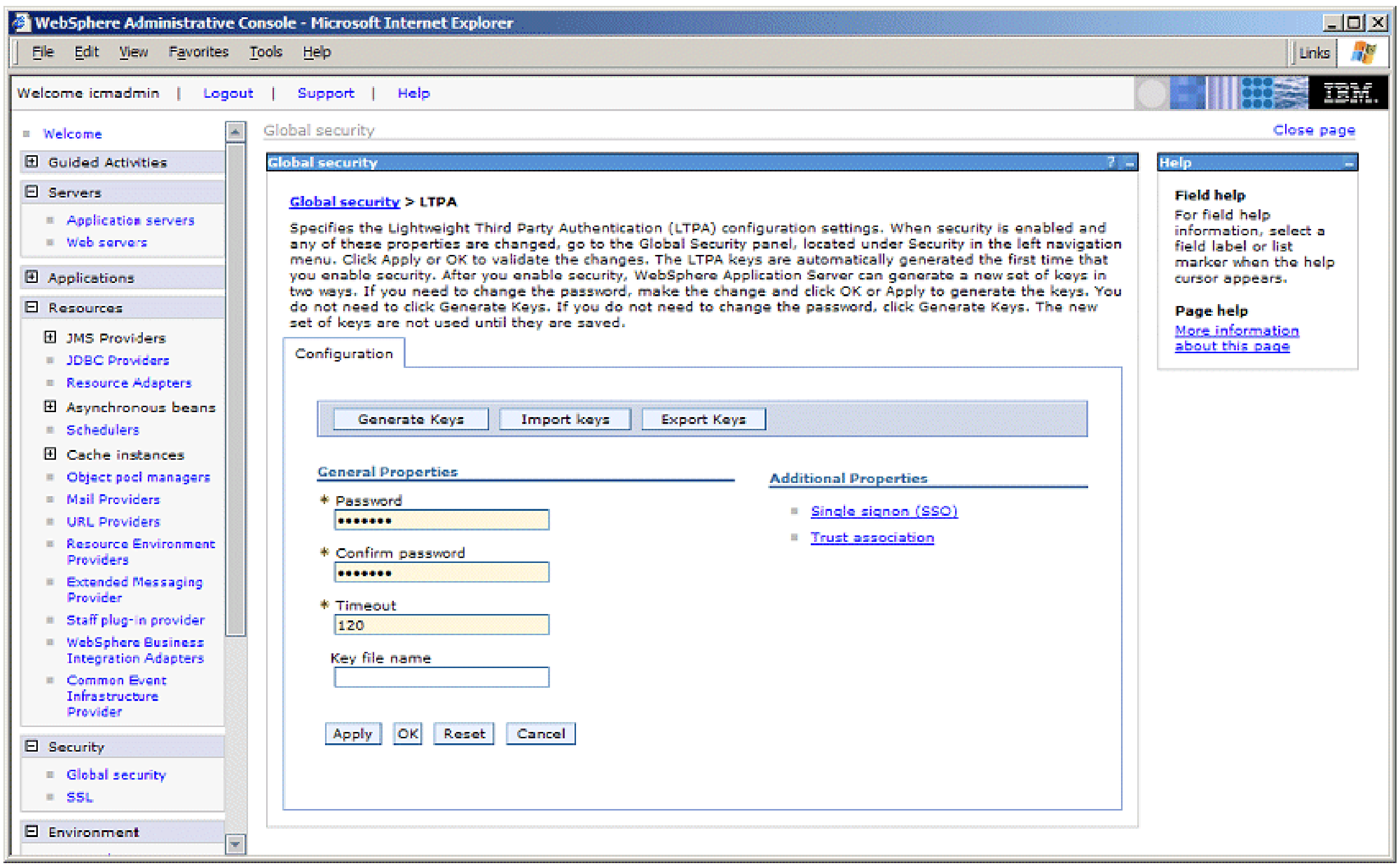
Select	Name	Value	Description
☐	groupsFile	${USER_INSTALL_ROOT}/security/groups.prop	
☐	usersFile	${USER_INSTALL_ROOT}/security/users.prop	
Total 2			

Help

Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
More information about this page

11. In the **Authentication** section of the **Security > Global Security** page select **Authentication mechanisms> LTPA**.
12. Enter a password into the **Password** and **Confirm password** fields (for example `icmadmin`'s password).



WebSphere Administrative Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help | Links

Welcome icmadmin | Logout | Support | Help

Welcome
Guided Activities
Servers
Application servers
Web servers
Applications
Resources
JMS Providers
JDBC Providers
Resource Adapters
Asynchronous beans
Schedulers
Cache instances
Object pool managers
Mail Providers
URL Providers
Resource Environment Providers
Extended Messaging Provider
Staff plug-in provider
WebSphere Business Integration Adapters
Common Event Infrastructure Provider
Security
Global security
SSL
Environment

Global security | Close page

Global security

Global security > **LTPA**

Specifies the Lightweight Third Party Authentication (LTPA) configuration settings. When security is enabled and any of these properties are changed, go to the Global Security panel, located under Security in the left navigation menu. Click Apply or OK to validate the changes. The LTPA keys are automatically generated the first time that you enable security. After you enable security, WebSphere Application Server can generate a new set of keys in two ways. If you need to change the password, make the change and click OK or Apply to generate the keys. You do not need to click Generate Keys. If you do not need to change the password, click Generate Keys. The new set of keys are not used until they are saved.

Configuration

Generate Keys | Import keys | Export Keys

General Properties

* Password: •••••••

* Confirm password: •••••••

* Timeout: 120

Key file name:

Apply | OK | Reset | Cancel

Additional Properties

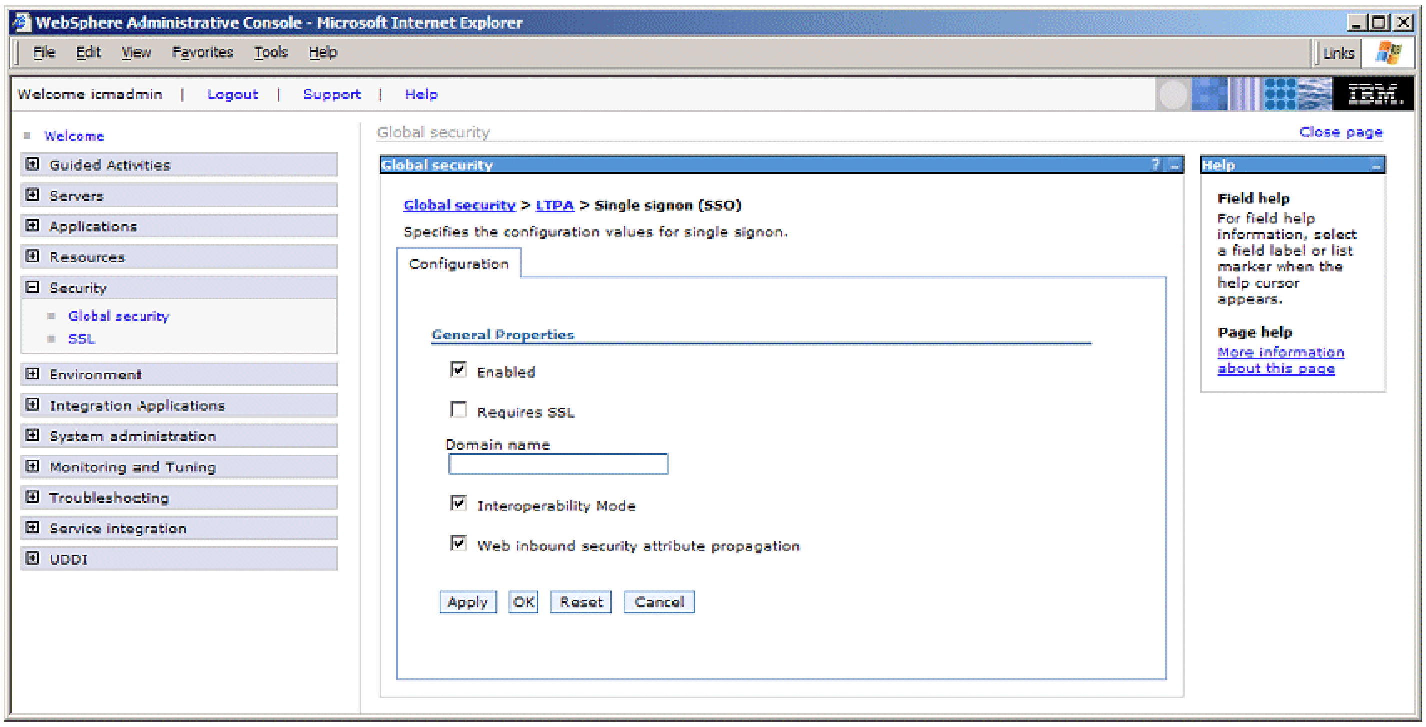
- Single signon (SSO)
- Trust association

Help

Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
More information about this page

13. In the **Additional Properties** section to the right click **Single signon (SSO)** and ensure that the **Enabled** box is checked.


WebSphere Administrative Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links
Welcome icmadmin | Logout | Support | Help
Welcome
Guided Activities
Servers
Applications
Resources
Security
Global security
SSL
Environment
Integration Applications
System administration
Monitoring and Tuning
Troubleshooting
Service integration
UDDI
Global security
Close page
Help
Field help
For field help information, select a field label or list marker when the help cursor appears.
Page help
More information about this page
Apply
OK
Reset
Cancel

Global security

Global security > LTPA > Single signon (SSO)

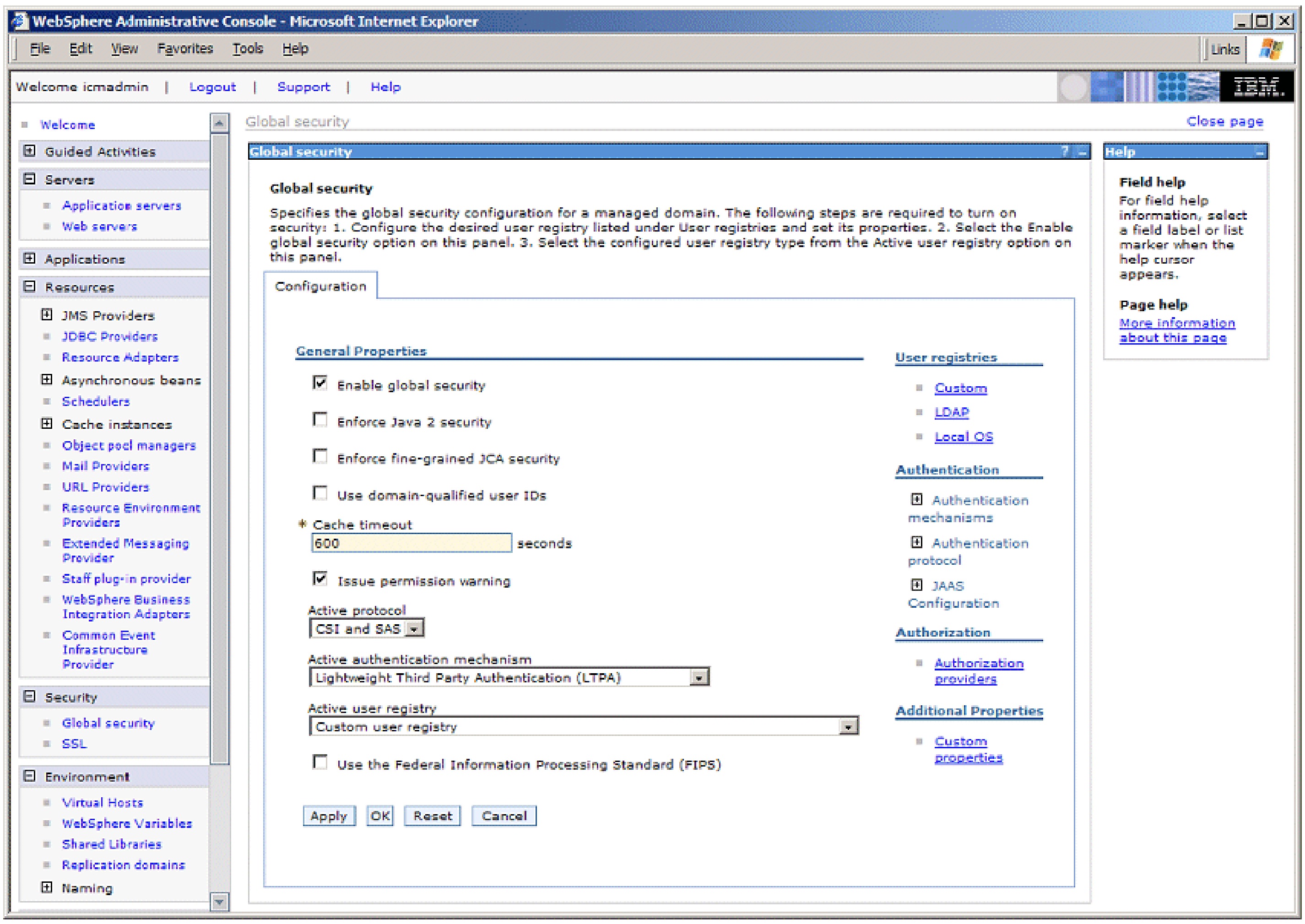
Specifies the configuration values for single signon.

Configuration

General Properties

- [✓] Enabled
- [ ] Requires SSL
- Domain name:
- [✓] Interoperability Mode
- [✓] Web inbound security attribute propagation

14. Click **OK** on the **Single signon** page and **OK** on the **Global security > LTPA** page to store the changes
15. On the **Security > Global Security** panel check the **Enabled** check box and uncheck the **Enforce Java 2 Security** check box.



Global security

Specifies the global security configuration for a managed domain. The following steps are required to turn on security: 1. Configure the desired user registry listed under User registries and set its properties. 2. Select the Enable global security option on this panel. 3. Select the configured user registry type from the Active user registry option on this panel.

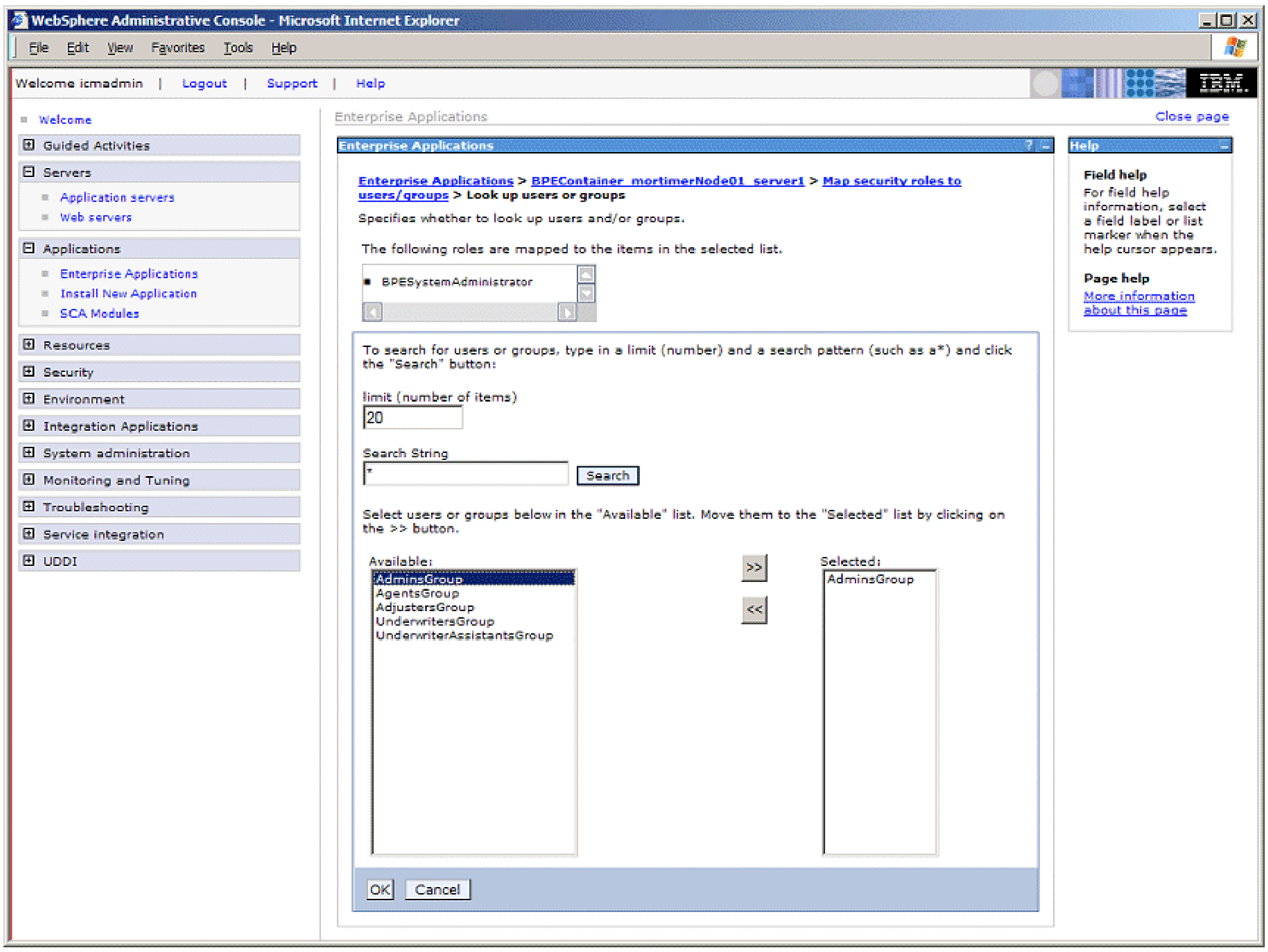
Configuration

General Properties

- [✓] Enable global security
- [ ] Enforce Java 2 security
- [ ] Enforce fine-grained JCA security
- [ ] Use domain-qualified user IDs
- * Cache timeout: 600 seconds
- [✓] Issue permission warning
- Active protocol: CSI and SAS
- Active authentication mechanism: Lightweight Third Party Authentication (LTPA)
- Active user registry: Custom user registry
- [ ] Use the Federal Information Processing Standard (FIPS)

16. Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.
17. Stop **server1** as described in section 6.2.

18. Click the **Servers** tab in WebSphere Integration Developer and double click your test server entry to open the `Server Overview` editor. Expand the **Security** section, check **Security is enabled on this server** and enter `icmadmin` and the corresponding password (you may need to collapse and re-expand the section to see the input fields). Press CTRL-S to store the changes.
19. Start **server1** (see 6.2 for details).
20. If you have installed the integrated test environment (you are not using a stand-alone installation of the WebSphere Process Server) you need to perform the following steps to enable it for use with security:
    - In the navigation panel select **Security > Global Security**. In the **Authentication** section expand **JAAS Configuration** and click **J2C Authentication data**.
    - For all predefined Authentication aliases which contain the user ID `wid`, change the user ID and password to `icmadmin` and the corresponding password.
    - In the navigation panel select **Applications > Enterprise Applications**.
    - Click the application with a name similar to `BPEContainer_widNode_server1`.
    - In the **Additional Properties** section click **Map security roles to users/roles**.
    - Check the box left of `BPESystemAdministrator` and click **Look up groups.**
    - Remove the group `wid` and add `AdminsGroup` to the list and click **OK**



Enterprise Applications > BPEContainer_mortimerNode01_server1 > Map security roles to users/groups > Look up users or groups

Specifies whether to look up users and/or groups.

The following roles are mapped to the items in the selected list.

BPESystemAdministrator

To search for users or groups, type in a limit (number) and a search pattern (such as a*) and click the "Search" button:

limit (number of items): 20

Search String: *

Select users or groups below in the "Available" list. Move them to the "Selected" list by clicking on the >> button.

Available: AdminsGroup, AgentsGroup, AdjustersGroup, UnderwritersGroup, UnderwriterAssistantsGroup

Selected: AdminsGroup

For field help information, select a field label or list marker when the help cursor appears.

- Perform the same sequence of steps with the enterprise application `TaskContainer_widNode_server1` (the role name is `TaskSystemAdministrator`.
- Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.

2.9 Importing, deploying, and running the sample

1. In WebSphere Integration Developer select **File > Import** from the menu. The **Import** window opens. Select **Project Interchange** and click **Next**. In the **From zip file** section click **Browse** to locate `<TMP>\projects.zip` and click **OK**. A list of 8 projects (`ClaimsHandling`, `CMBPCIntegration`, `CMBPCIntegrationBeans`, `CMBPCIntegrationLibrary`, `CMConnectionManagement`, `ContentViewerApplet`, `QuickStartClient`, `QuickStartClientApp`) shows up in the section below. Click **Select All** and **Finish** to import the projects.

2. In the **Project Explorer** view of the **J2EE** perspective expand **EJB Projects > CMBPCIntegrationBeans** and double-click the **Deployment Descriptor**
3. In the **Overview** tab of the deployment descriptor editor locate and expand the section **Ejb Client Jar**. Click the **Create** button. Accept the proposed name `CMBPCIntegrationBeansClient` in the **EJB client JAR Creation** dialogue and click **Finish** to start project creation.
4. Note that the creation of the client project has updated the build path of the corresponding server project. This raises a number of build path errors that need to be fixed as follows: in the **EJB Projects** section of the **Project Explorer** view right-click **CMBPCIntegrationBeans** and select **Properties**. Click **Java Build Path**, switch to the **Projects** tab and check **CMBPCIntegration**. Click **OK** to close the properties window and verify that the exceptions are gone.
5. In the **Project Explorer** view expand **Enterprise Applications > CMBPCIntegrationApp** and double click the **Deployment Descriptor**
6. In the Application Deployment Descriptor editor select the **Security** tab and click **Gather** to add the `CMAdministrator` role to the list. Click **CMAdministrator** and check **Users/Groups** in the **WebSphere Bindings** section. Scroll down to the **Groups** section and add the administrator group of this system (to which `icmadmin` belongs, on a Windows system typically `Administrators`) to the list by clicking **Add** entering the group name and clicking **Finish**. Scroll down to the **Security Role Run As Bindings** section and add `icmadmin` and the corresponding password. Press CTRL-S to save your changes.
7. Note that the **Problems** view lists about 100 warnings concerning unused imports and calls to deprecated methods. These refer to the generated resources and can safely be ignored.
8. Perform the steps described in 6.6 to deploy the three enterprise applications on the test server.

You are now ready to explore the sample process as described in the next chapter.

3. A Tour of the 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 DB2 Content Manager).

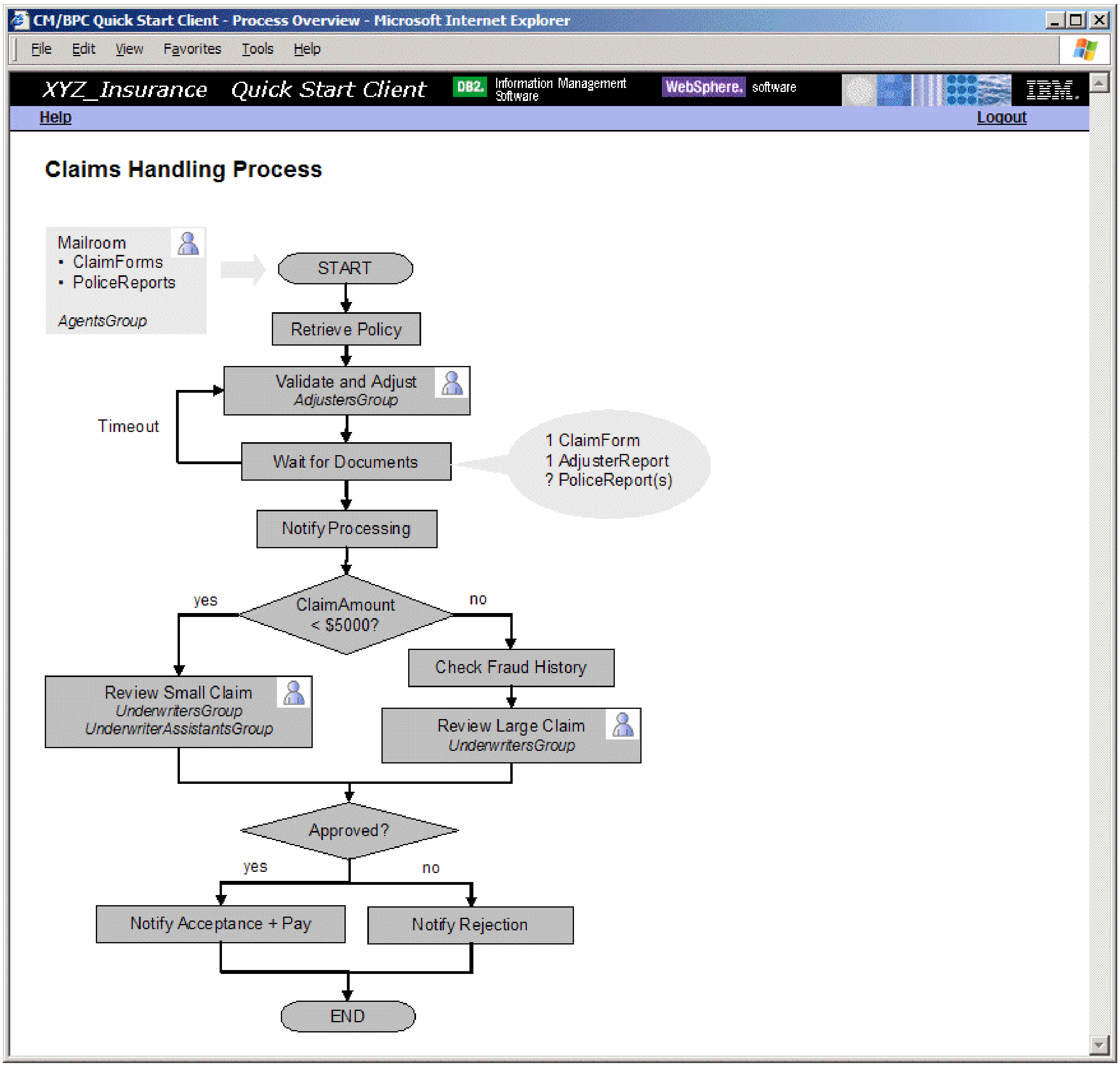
The new sample process provided with CM/WPS Integration Quick Start focuses on the integration of different services both human-based and automatic. Automated services such as retrieval of the insurance policy or the fraud history check illustrate the inclusion of third party applications or B2B transactions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

This is a more natural way of demonstrating the use of WebSphere Process Server since the DB2 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/WPS 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.

- A link similar to (depending on the port your server is using): **http://localhost:9080/QuickStartClient** opens 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.



- The boxes that have a schematic representation of a person in their top right corner represent staff activities. The other boxes represent (automated) services. Diamonds represent decisions and the rounded rectangles indicate the beginning and end of a process instance's lifetime (note that the sample process is configured such that process instances are deleted upon termination).
- 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.
- The process enclosed by rounded rectangles consists of three sections: 1. data validation, 2. review, 3. completion.

1. **Data validation** ensures that the information needed to review the case is accurate and complete. It consists of an automated step (Retrieve Policy), 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 needed for the review 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 are stored 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 period of time. 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 perform appropriate actions by

requesting the required documents or weakening the collection point condition. When the collection point condition is passed, the communication service (that keeps track of the communication with external parties) sends a notification out to the client.

2. The **Review** section of the process begins with a **Decision Point** (switch activity in BPEL terminology) where one of two available paths is taken depending on the size of the claim expressed in an estimated overall cost that is part of the adjuster report (`ClaimAmount` attribute). For small claims, members of the UnderwriterAssistantsGroup investigate the documents in the folder and complete the review activity by approving or rejecting the claim. With a `ClaimAmount` > $5000 claims are considered big. In this case, an external **FraudHistory** service is consulted that prepares a fraud history report assessing the trustworthiness of the claimant based on certain criteria derived from the case and policy data. The fraud history report is made available to members of the UnderwritersGroup who are entitled to review either small or large claims. However, only members of the UnderwritersGroup can approve or reject large claims.
3. The **Completion** section of the process sends an approval or rejection letter back to the client depending on the outcome of the review. In case of approval it triggers a payment transaction that runs in parallel to the notification.

- 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:

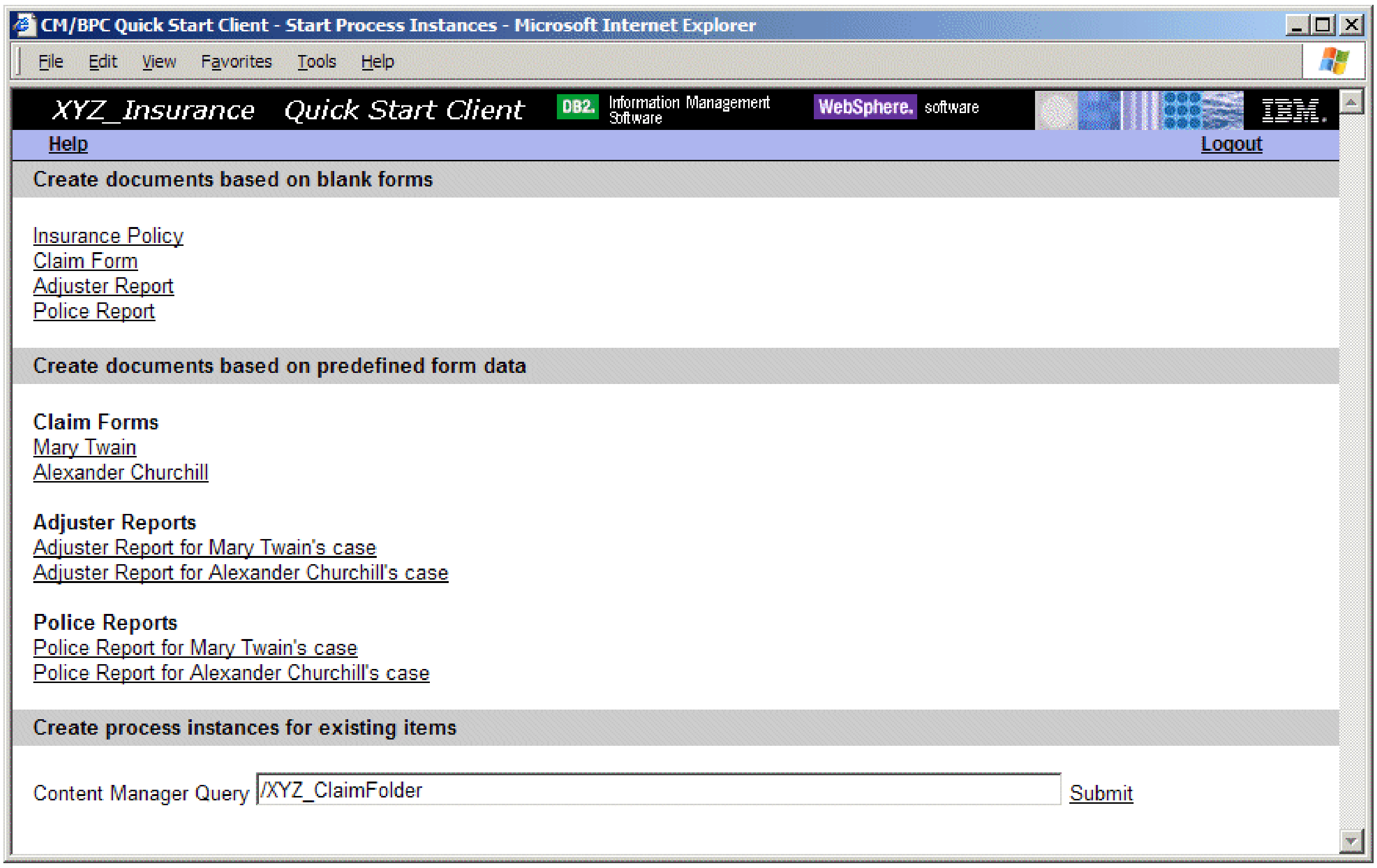
	Group	Group members
Mailroom	`AgentsGroup`	`Agent1, Agent2`
ValidateAndAdjust	`AdjustersGroup`	`Adjuster1, Adjuster2`
ReviewSmallClaim	`UnderwriterAssistantsGroup`	`UnderwriterAssistant1, UnderwriterAssistant2`
ReviewLargeClaim	`UnderwritersGroup`	`Underwriter1, Underwriter2`

If your library server name is not `icmnlsdb` 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 two different ways to create process instances either explicitly by an application or implicitly based on Content Event Handling. Explicit creation 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 can serve as a basis for custom development. Implicit creation 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.

1. 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).
2. The Mailroom user's initial page contains three sections titled **Create documents based on blank forms**, **Create documents based on predefined form data**, and **Create process instances for existing items**


CM/BPC Quick Start Client - Start Process Instances - Microsoft Internet Explorer
File Edit View Favorites Tools Help
XYZ_Insurance Quick Start Client
DB2. Information Management Software
WebSphere. software
IBM.
Help
Logout

Create documents based on blank forms

Insurance Policy
Claim Form
Adjuster Report
Police Report

Create documents based on predefined form data

Claim Forms
Mary Twain
Alexander Churchill

Adjuster Reports
Adjuster Report for Mary Twain's case
Adjuster Report for Alexander Churchill's case

Police Reports
Police Report for Mary Twain's case
Police Report for Alexander Churchill's case

Create process instances for existing items

Content Manager Query /XYZ_ClaimFolder Submit

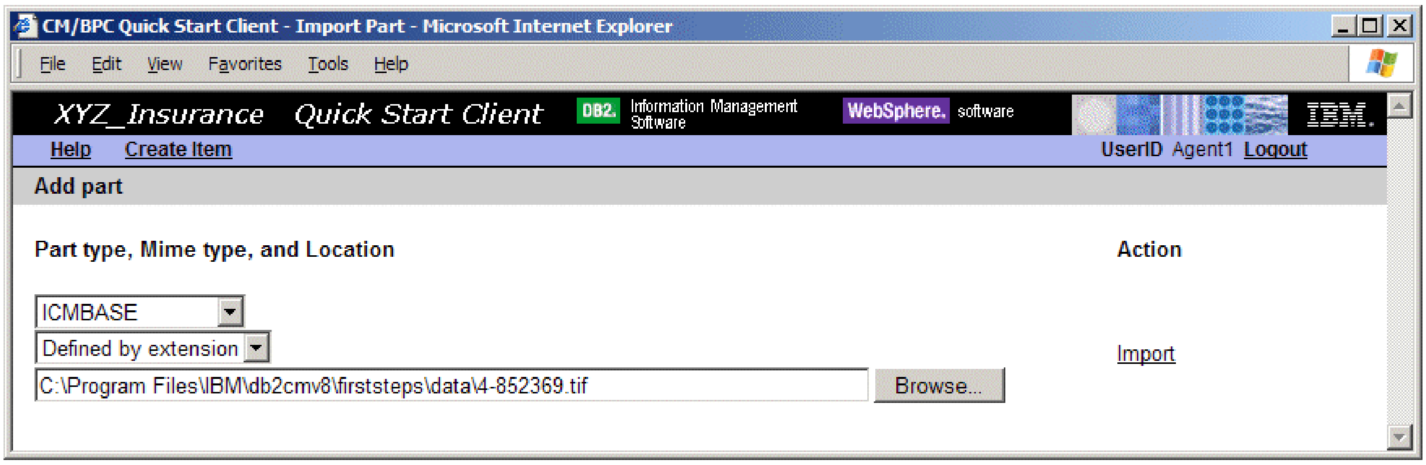
3. The section titled **Create documents based on blank forms** can be used to create instances of the four item types `XYZ_InsPolicy`, `XYZ_ClaimForm`, `XYZ_AdjReport`, and `XYZ_PolReport`. The latter three are linked to a `XYZ_ClaimFolder` that has the attribute `WF_OnCreate` with a default value of `ClaimsHandlingProcess`. Creating an item of type `XYZ_ClaimForm` triggers the creation of an item of type `XYZ_ClaimFolder` which contains it due to auto-foldering. The creation of this folder triggers the creation of an instance of the `ClaimsHandlingProcess` which carries a reference to the claim folder due to the `WF_OnCreate` attribute and its default value (the process name).
4. Since creating an item may involve entering many mandatory values, the section **Create documents based on predefined form data** can be used to create items of the three types based on pre-defined attribute values. These values make use of the DB2 Content Manager First Steps sample data and are expliticly defined in `startProcessInstances.jsp`. Click any of the links to open the corresponding form.
5. The section **Create process instances for existing items** can be used to create process instances based on the sample data already importet into DB2 Content Manager during the 'Create Sample Data' step of the DB2 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[@XYZ_ClaimNumber = "4-852369"]`) the query can be refined to restrict the items for which process instances should be created to e.g. all claim forms with claim number `4-852369`. Note that when starting processes for the DB2 Content Manager sample data based on a query you may see warnings in the console indicating that the policy can not be added to the folder since the folder already contains it. This is due to the fact that each folder of the sample data already contains instances of all three item types. The message can be ignored.

Import predefined documents

1. This tour starts with the section **Create documents based on predefined form data** since this illustrates the use of content event handling and is based on existing data.
2. Click **Mary Twain** in the section titled **Claim Forms** to display the attributes of a predefined claim form. Review the displayed values and make sure the claim number is a new unique value. If you re-use an already existing claim number the claim form will be added to an existing folder that is already part of a running process so no new process will be created (see 6.1 for details).
3. Click **Save** to store an instance of the `XYZ_ClaimForm` item type in DB2 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.
4. The console output of the test server confirms the creation of an item (`-> Item created`). If content event handling works properly, another message is displayed that says `=> Process started with ItemPID [...]`. Note that the

message prefix indicates its origin. Messages starting with `->` originate from the Quick Start Client. Those starting with `=>` result from the integration toolkit and if the prefix is `>>` they originate from the process. The messages `=> Looking up policy with number: ...` and `=> Number of matching insurance policies for claim [...]: 1)` indicate the successful completion of the first process activity (`RetrievePolicy`).

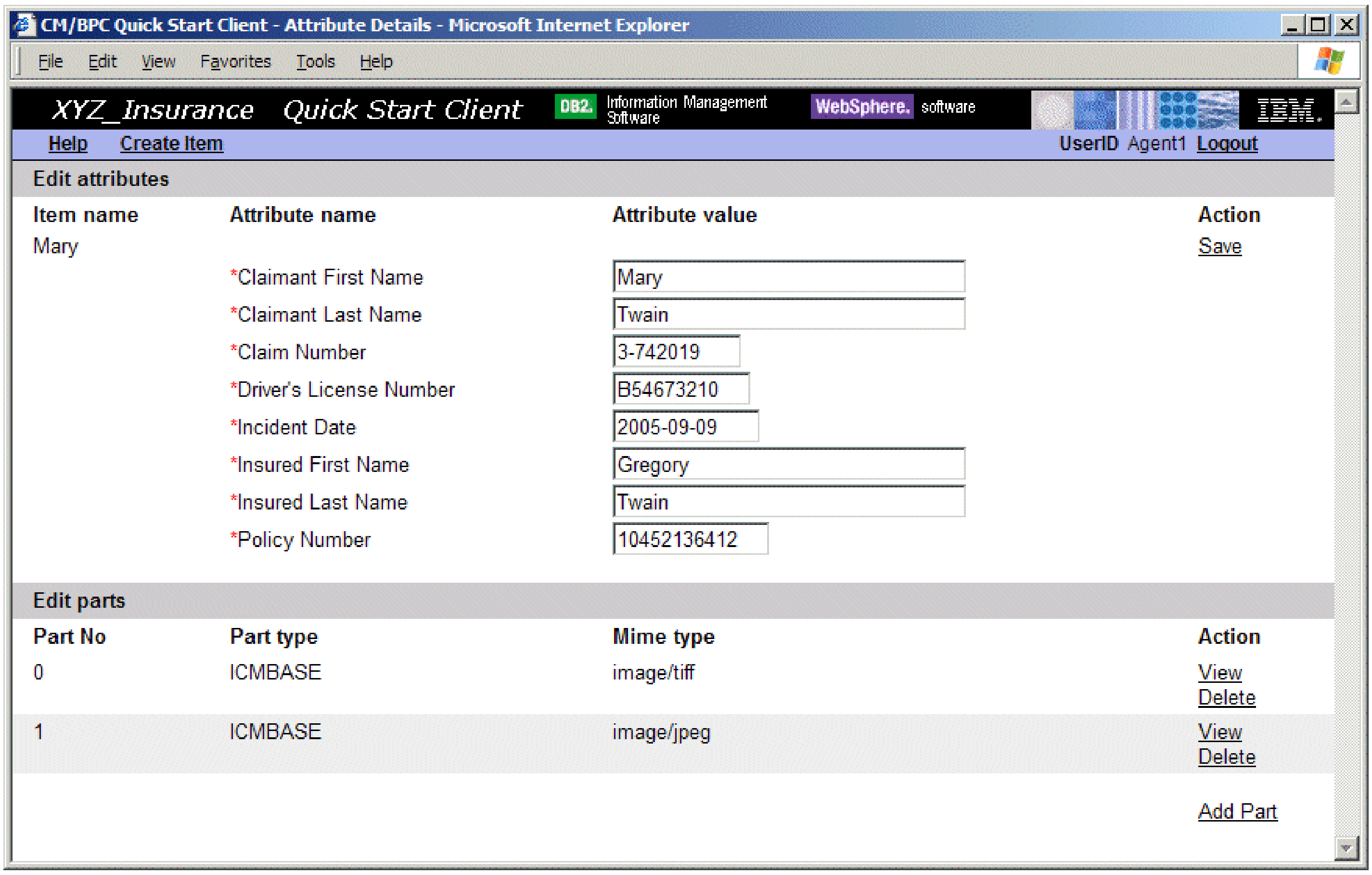
5. The client now displays an `Add part` link that can be used to attach the actual document. Clicking **Add part** switches to the **Import part** page.


CM/BPC Quick Start Client - Import Part - Microsoft Internet Explorer
File Edit View Favorites Tools Help
XYZ_Insurance Quick Start Client
DB2. Information Management Software
WebSphere. software
IBM.
Help Create Item
UserID Agent1 Logout
Import
Browse...

Add part

Part type, Mime type, and Location	Action
ICMBASE	
Defined by extension	Import
C:\Program Files\IBM\db2cmv8\firststeps\data\4-852369.tif	

6. Locate the TIFF image of a scanned claim form in `<IBMCMROOT>\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 may add another part by importing the incident photo `<IBMCMROOT>\firststeps\data\1ABC089.jpg`. Note that in this case you need to explicitly select the mime type `JPEG` from the list since automatic detection fails for files of this type.


CM/BPC Quick Start Client - Attribute Details - Microsoft Internet Explorer
File Edit View Favorites Tools Help
XYZ_Insurance Quick Start Client
DB2. Information Management Software
WebSphere. software
IBM.
Help Create Item
UserID Agent1 Logout

Edit attributes

Item name	Attribute name	Attribute value	Action
Mary			Save
	*Claimant First Name	Mary	
	*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	

Edit parts

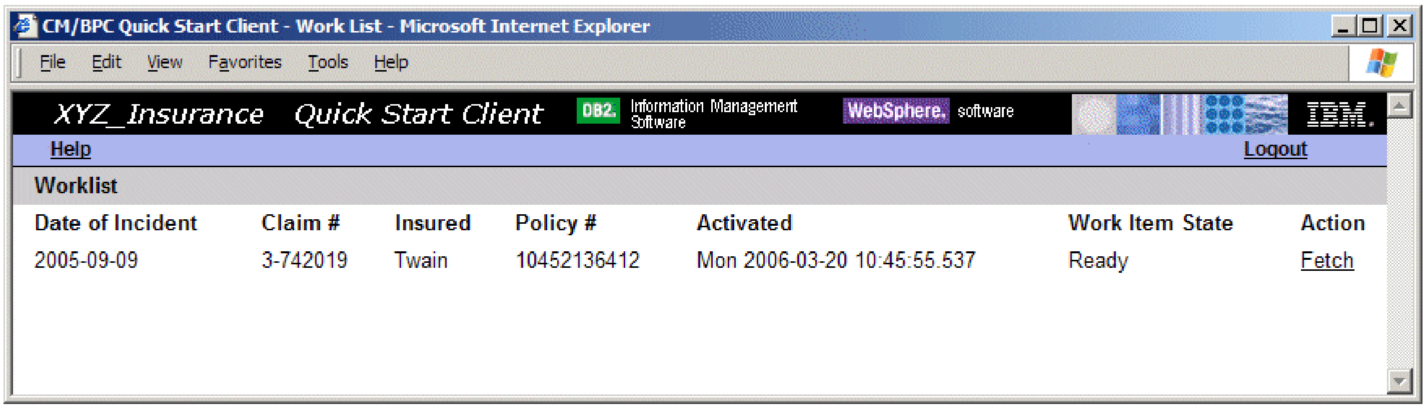
Part No	Part type	Mime type	Action
0	ICMBASE	image/tiff	View Delete
1	ICMBASE	image/jpeg	View Delete
			Add Part

7. Clicking **Create item** in the menu leads back to the Mailroom home page from which further items can be created. For now we leave the Mailroom dialogue by clicking **Logout** at the right end of the client menu.

3.3 Data validation

The message => `Number of matching insurance policies for claim [...]: 1` in the server console confirms that an appropriate insurance policy has been located in DB2 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.

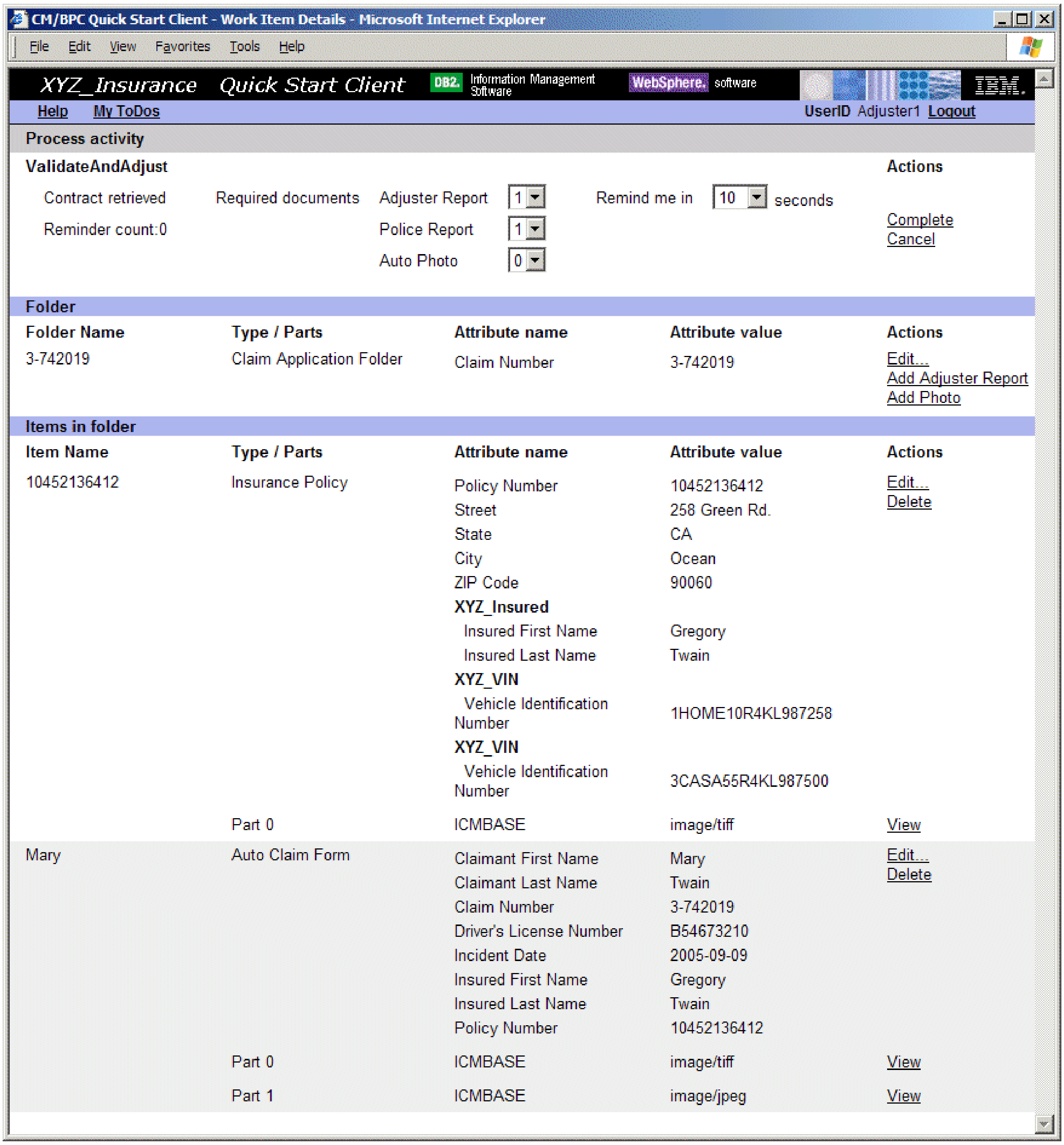
1. On the process diagram click the box **Validate and Adjust** and log on as `Adjuster1`.
2. The adjuster's initial page shows the adjuster's worklist. It contains a single work item corresponding to the process instance that has been created in the previous step (Mary Twain's claim form). The worklist 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 DB2 Content Manager query in the form >> `Query [/XYZ_ClaimFolder[@ITEMID IN ("...")]]`


CM/BPC Quick Start Client - Work List - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
XYZ_Insurance Quick Start Client
DB2. Information Management Software
WebSphere. software
IBM.
Help
Logout

Worklist

Date of Incident	Claim #	Insured	Policy #	Activated	Work Item State	Action
2005-09-09	3-742019	Twain	10452136412	Mon 2006-03-20 10:45:55.537	Ready	Fetch

3. 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`. Note that `Adjuster1` now owns this work item. If another member of the `AdjustersGroup` runs a work list query by opening his work list page, this work item will not show up any longer.



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

File Edit View Favorites Tools Help

XYZ_Insurance Quick Start Client

Help My ToDos UserID Adjuster1 Logout

Process activity

ValidateAndAdjust **Actions**

Contract retrieved	Required documents	Adjuster Report	1	Remind me in 10 seconds	Complete Cancel
Reminder count:0		Police Report	1		
		Auto Photo	0		

Folder

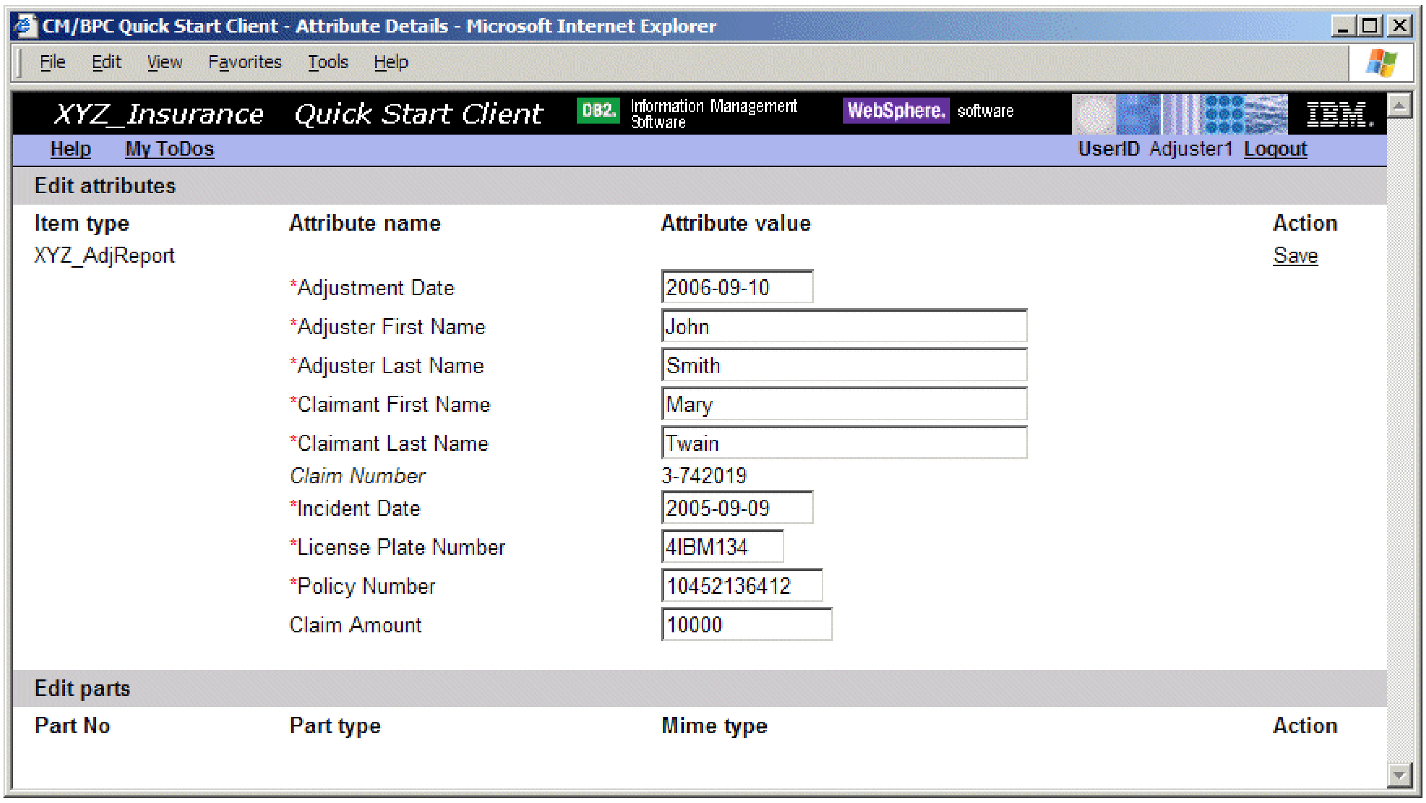
Folder Name	Type / Parts	Attribute name	Attribute value	Actions
3-742019	Claim Application Folder	Claim Number	3-742019	Edit... Add Adjuster Report Add Photo

Items in folder

Item Name	Type / Parts	Attribute name	Attribute value	Actions
10452136412	Insurance Policy	Policy Number	10452136412	Edit... Delete
		Street	258 Green Rd.	
		State	CA	
		City	Ocean	
		ZIP Code	90060	
		XYZ_Insured		
		Insured First Name	Gregory	
		Insured Last Name	Twain	
		XYZ_VIN		
		Vehicle Identification Number	1HOME10R4KL987258	
		XYZ_VIN		
		Vehicle Identification Number	3CASA55R4KL987500	
	Part 0	ICMBASE	image/tiff	View
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	
	Part 0	ICMBASE	image/tiff	View
	Part 1	ICMBASE	image/jpeg	View

4. The work item page contains three sections titled **Process activity**, **Folder**, and **Items in Folder**. The **Process activity** section basically displays the input 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.
5. 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.
6. 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.

7. 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 DB2 Content Manager.


CM/BPC Quick Start Client - Attribute Details - Microsoft Internet Explorer
File Edit View Favorites Tools Help
XYZ_Insurance Quick Start Client
Help My ToDos
UserID Adjuster1 Logout

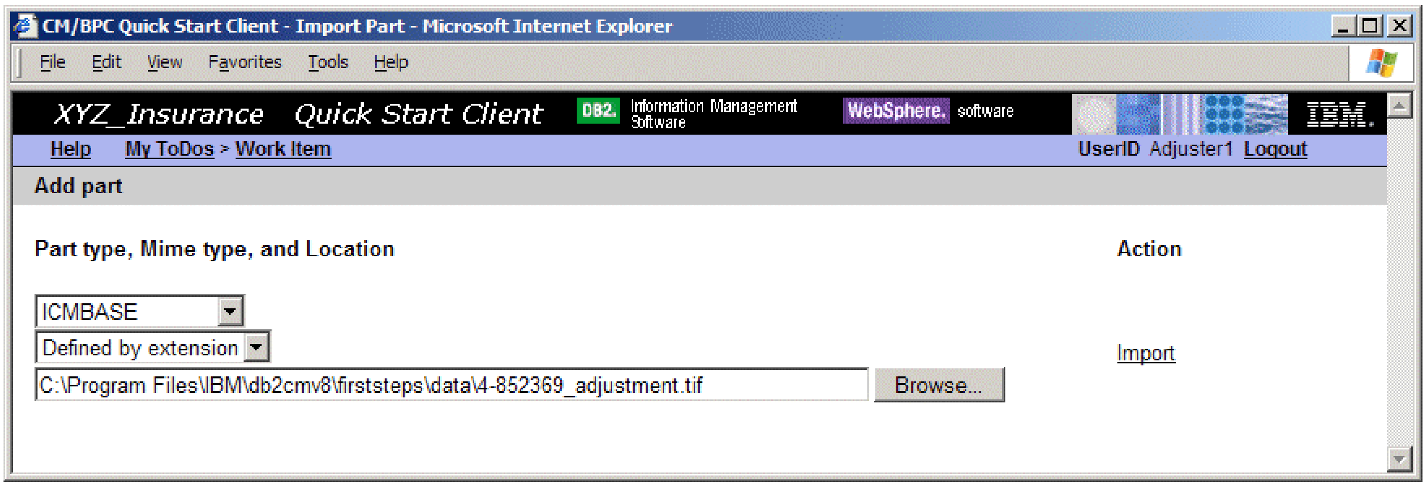
Edit attributes

Item type	Attribute name	Attribute value	Action
XYZ_AdjReport			Save
	*Adjustment Date	2006-09-10	
	*Adjuster First Name	John	
	*Adjuster Last Name	Smith	
	*Claimant First Name	Mary	
	*Claimant Last Name	Twain	
	Claim Number	3-742019	
	*Incident Date	2005-09-09	
	*License Plate Number	4IBM134	
	*Policy Number	10452136412	
	Claim Amount	10000	

Edit parts

Part No	Part type	Mime type	Action

8. Click **Add part** and import `<IBMCMROOT>\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.

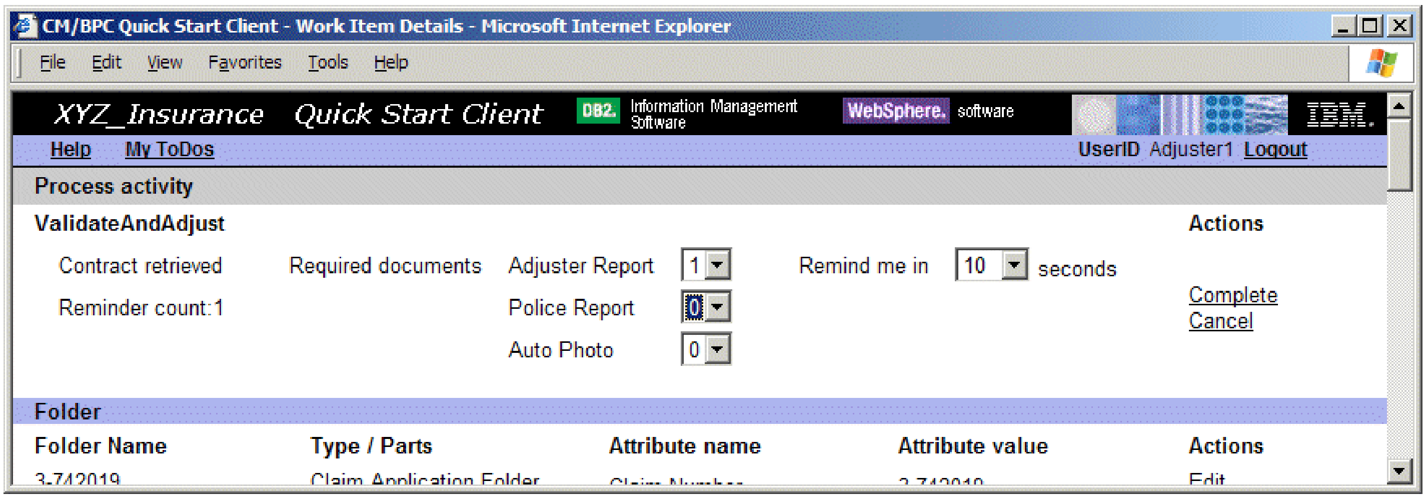

CM/BPC Quick Start Client - Import Part - Microsoft Internet Explorer
File Edit View Favorites Tools Help
XYZ_Insurance Quick Start Client
Help My ToDos > Work Item
UserID Adjuster1 Logout
Browse...

Add part

Part type, Mime type, and Location — Action

ICMBASE
Defined by extension
C:\Program Files\IBM\db2cmv8\firststeps\data\4-852369_adjustment.tif — Import

9. 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.
10. 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`.
11. 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.
12. 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.



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

XYZ_Insurance Quick Start Client

Help My ToDos UserID Adjuster1 Logout

Process activity

ValidateAndAdjust

					Actions
Contract retrieved	Required documents	Adjuster Report	1	Remind me in 10 seconds	Complete
Reminder count:1		Police Report	0		Cancel
		Auto Photo	0		

Folder

Folder Name	Type / Parts	Attribute name	Attribute value	Actions
3-742019	Claim Application Folder	Claim Number	3-742019	Edit

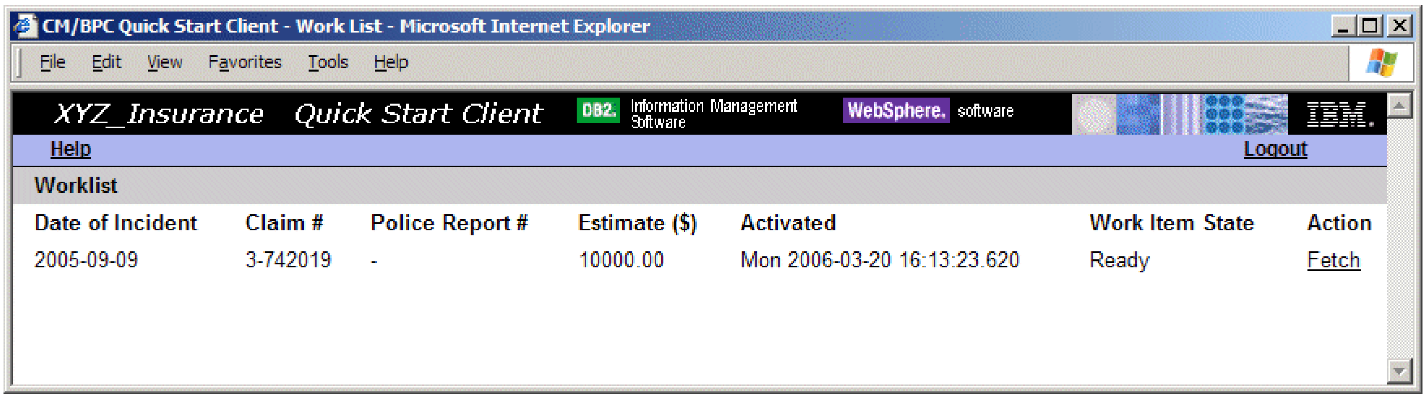
13. Click **Complete** to complete the adjuster's activity and forward the work item to the next step in the process. The console output displays the message `-> Work item completed` to confirm successful completion of this step.
14. Since Adjuster1 has changed the number of required Police Reports to 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: 3-742019` and a confirmation is sent to the person that submitted the claim (`=> Dear...`). Furthermore, the following two messages are shown: `=> 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.
15. Log out Adjuster1 to return to the process overview.

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.

1. 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.



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

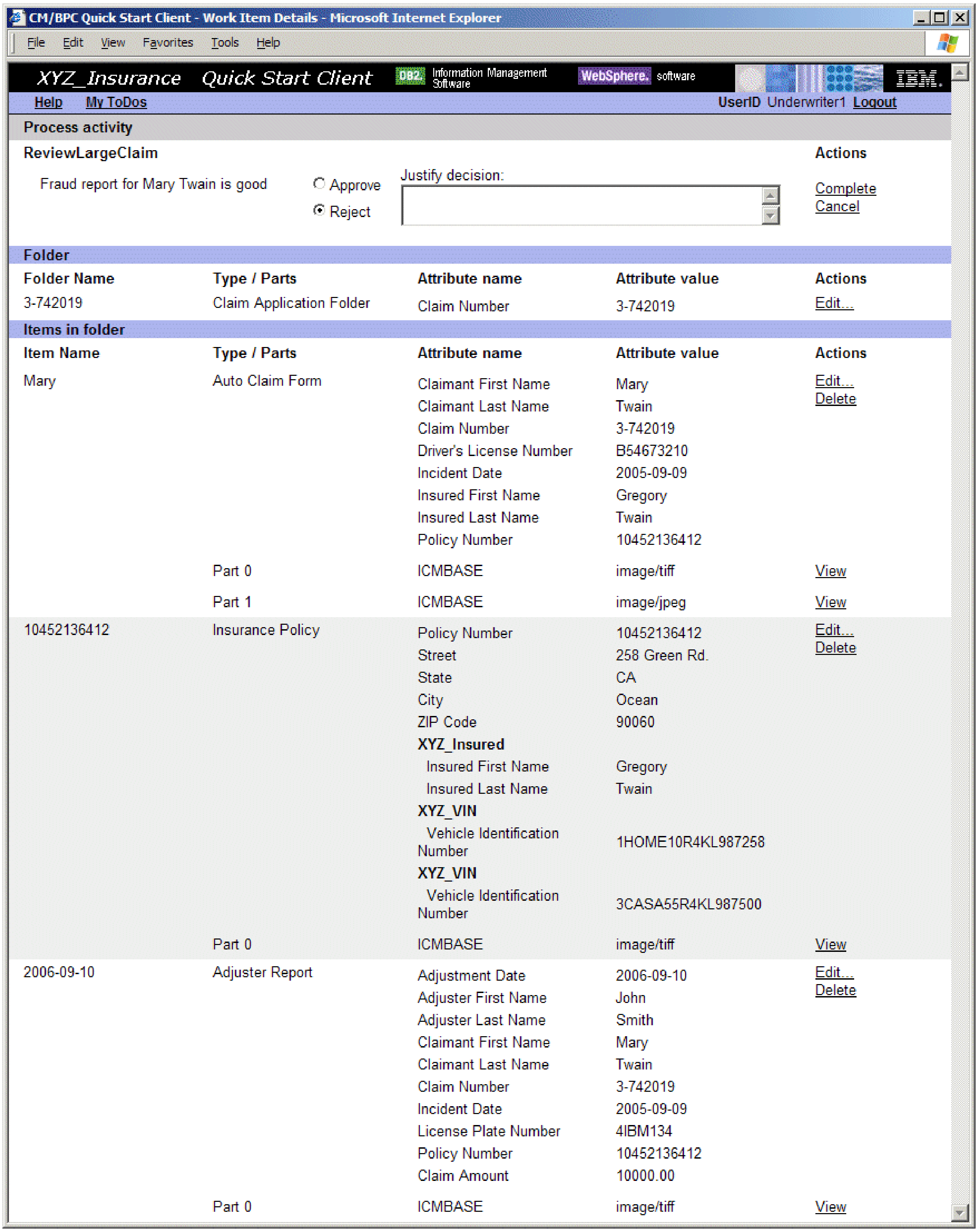
XYZ_Insurance Quick Start Client

Help Logout

Worklist

Date of Incident	Claim #	Police Report #	Estimate ($)	Activated	Work Item State	Action
2005-09-09	3-742019	-	10000.00	Mon 2006-03-20 16:13:23.620	Ready	Fetch

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


CM/BPC Quick Start Client - Work Item Details - Microsoft Internet Explorer
File Edit View Favorites Tools Help
XYZ_Insurance Quick Start Client
Help My ToDos
UserID Underwriter1 Logout

Process activity

ReviewLargeClaim

Fraud report for Mary Twain is good

Approve [✓ Reject]

Justify decision:

Actions: Complete, Cancel

Folder

Folder Name	Type / Parts	Attribute name	Attribute value	Actions
3-742019	Claim Application Folder	Claim Number	3-742019	Edit...

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	
	Part 0	ICMBASE	image/tiff	View
	Part 1	ICMBASE	image/jpeg	View
10452136412	Insurance Policy	Policy Number	10452136412	Edit... Delete
		Street	258 Green Rd.	
		State	CA	
		City	Ocean	
		ZIP Code	90060	
		XYZ_Insured		
		Insured First Name	Gregory	
		Insured Last Name	Twain	
		XYZ_VIN		
		Vehicle Identification Number	1HOME10R4KL987258	
		XYZ_VIN		
		Vehicle Identification Number	3CASA55R4KL987500	
	Part 0	ICMBASE	image/tiff	View
2006-09-10	Adjuster Report	Adjustment Date	2006-09-10	Edit... Delete
		Adjuster First Name	John	
		Adjuster Last Name	Smith	
		Claimant First Name	Mary	
		Claimant Last Name	Twain	
		Claim Number	3-742019	
		Incident Date	2005-09-09	
		License Plate Number	4IBM134	
		Policy Number	10452136412	
		Claim Amount	10000.00	
	Part 0	ICMBASE	image/tiff	View

3. 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 DB2 Content Manager. Thus it is available throughout the process and potentially stored in the process log but not made persistent in the content repository.
4. Click **View** in the Actions column for the Auto Claim Form and investigate the annotations which the Adjuster has added to the document.
5. Type OK into the field **Justify decision**, select **Approve**, and click **Complete**.

6. The console output of the test server now displays the message `=;> Dear Mr. Twain, your claim (3-742019) has been accepted. <Reason: OK>. Yours Sincerely XYZ_Insurance.` Note that `<Reason: ...<` corresponds to a text module which might be rendered into an acceptable text by the communication service.
7. `=> Pay claim subprocess started on item [...] [...] [...]` indicate a subprocess that performs the corresponding financial transaction in parallel to the notification. Note that in place of a real BPEL process we simulate this activity by a simple Java class that implements the `CMProcessInterface` and displays this message.

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 `=> Dear Mr. Twain, we apologize that your claim(...) has been rejected. <Reason: ...>....`
- Note that the availibility 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.

3.6 How the Sample Process is defined

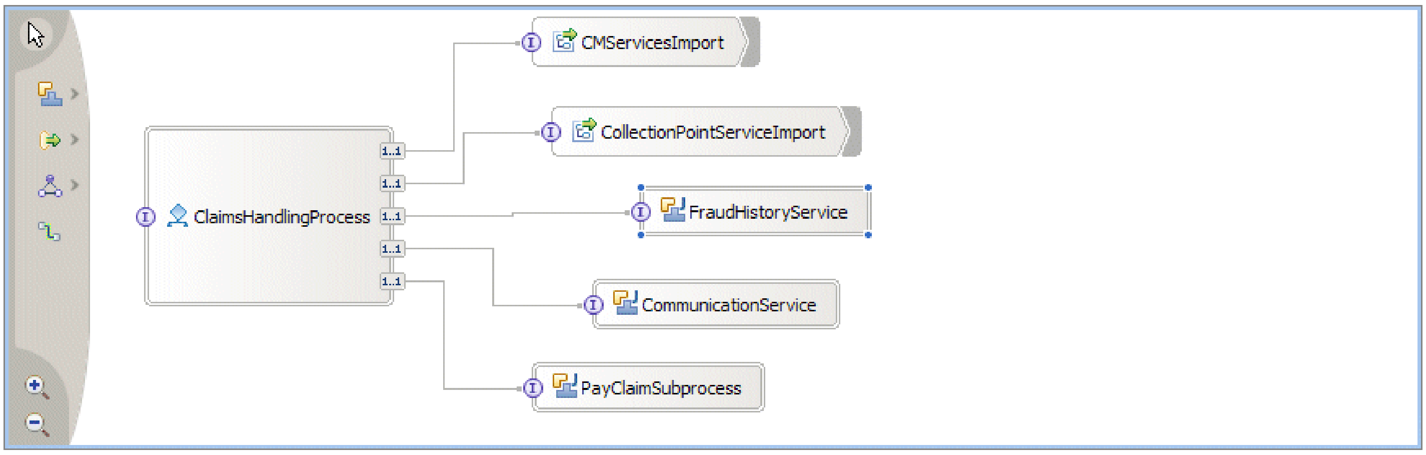
This section points to some important concepts and artifacts in WebSphere Integration Developer and explains their role for the sample process.

Dependencies

- Right-click the `ClaimsHandling` module (the module hosting the sample process) and select **Open Dependency Editor**. Note that this module depends on the library `CMBPCIntegrationLibrary` which is provided by the Integration Toolkit. This library contains interface, business object, and Java definitions needed when working with the functions of the Integration Toolkit.
- Close the project dependency editor.
- Expand `CMBPCIntegrationLibrary` and note that it consists of definitions for Data Types (business objects) and Interfaces. In the **Physical Resources** view (**Window > Show View > Physical Resources**) you can see that this library also contains a Java interface `com\ibm\bpe\cm\CMBPCConstants`

The Assembly Diagram

- In the Business Integration view expand the `ClaimsHandling` module und double click `ClaimsHandling` right below. This opens the **Assembly Diagram** of the `ClaimsHandling` SCA component in the assembly editor



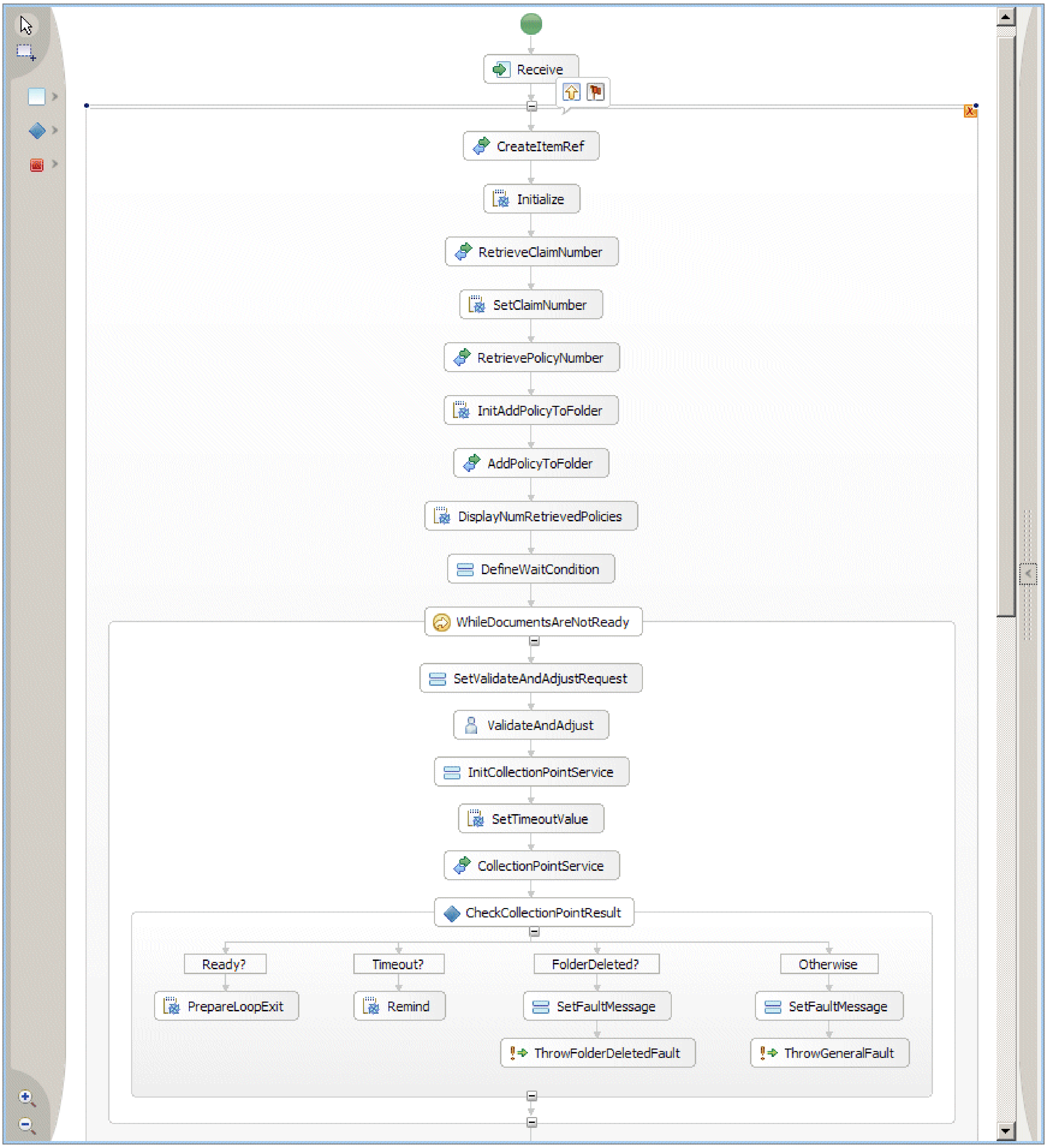
- Note that the `ClaimsHandlingProcess` component is wired to five supplementary services. The first two (`CMServicesImport`, `CollectionPointServiceImport`) are imported from the Integration Toolkit. The other three (`FraudHistoryService`, `CommunicationService`, `PayClaimSubprocess`) are specific to the sample process. For the sake of keeping the sample process simple and concentrating on the integration with DB2 Content Manager, these services have a simple Java implementation that can be found in the `com\ibm\bpe\cm\sca\` folder of this module

(Physical Resources view). Double-clicking a component's box in the assembly diagram opens the corresponding implementation if it is in the same module. In case of Java implementations it opens the Java source page editor. In case of the process, it opens the BPEL definition of the process.

- The Communication Service that handles all external communication of the insurance company writes skeletons of the corresponding mails to the server's output console.
- The Fraud History Services decides on a client's trustworthiness based on the initial letter of the client's last name treating all fraud reports for clients who's last name starts with A-L as bad and the others as good -- not a real fair treatment but deterministic for demonstration purposes.
- As the other services, the PayClaimSubprocess is implemented by a Java class that prints a corresponding statement to the server's output console. Its name suggests that it would normally be implemented as a subprocess. If you click it and investigate the **Details** tab of the **Properties** view you can see that it implements the same interface (`CMProcessInterface`) as the `ClaimsHandlingProcess` so it would also be a content-centric process.

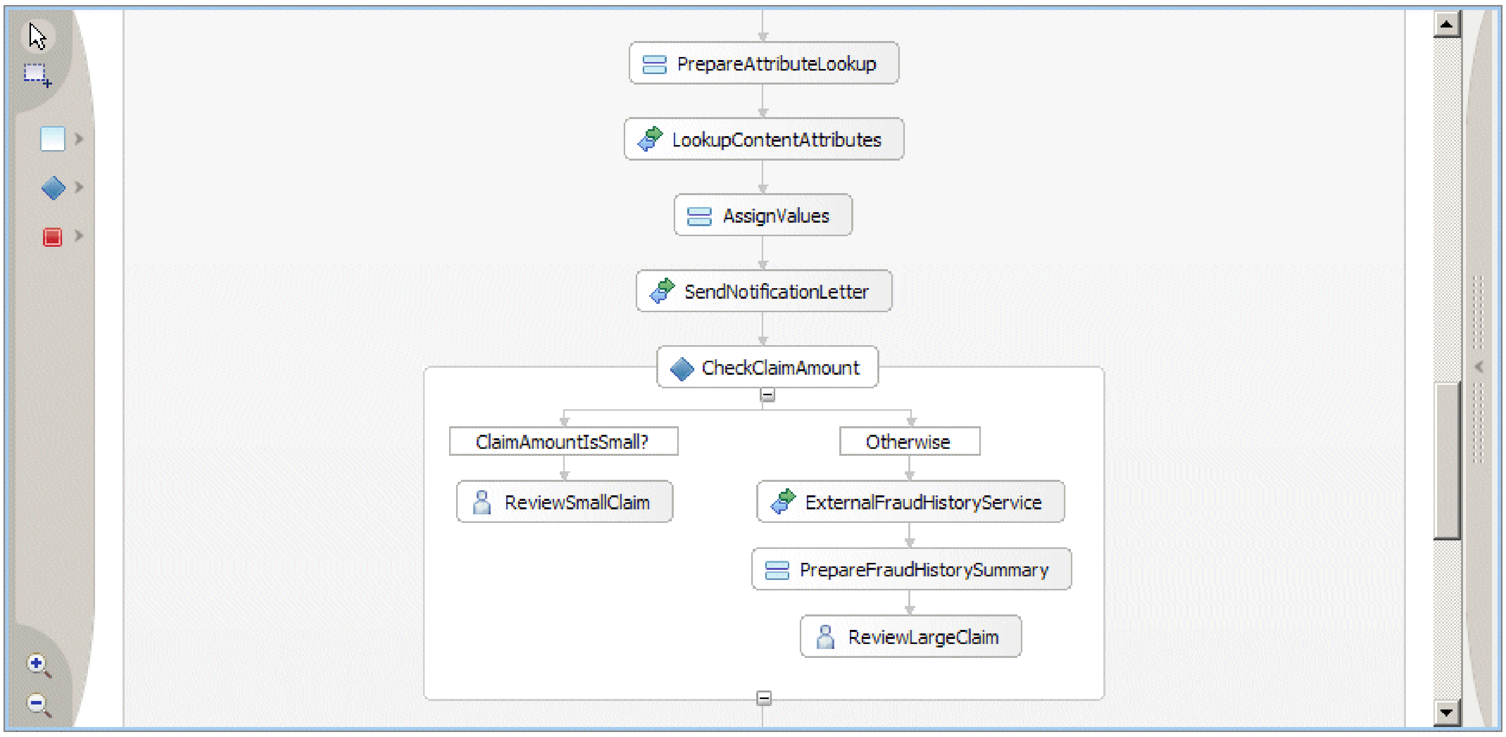
The Process Definition

- Double-click the box representing the `ClaimsHandlingProcess` in the assembly diagram. This opens the process definition in the BPEL editor.
- The process is defined on the basis of simple or structured activities represented as boxes. In the following we explain some of these activities and their role in the process. Clicking the box representing an activity and investigating the **Properties view** below the process diagram (especially the information under the **Details** tab) provides useful information about what the activity actually does how it is defined.
- The following screenshot shows the **Data Validation** segment of the process (according to the terminology used in the Overview section).

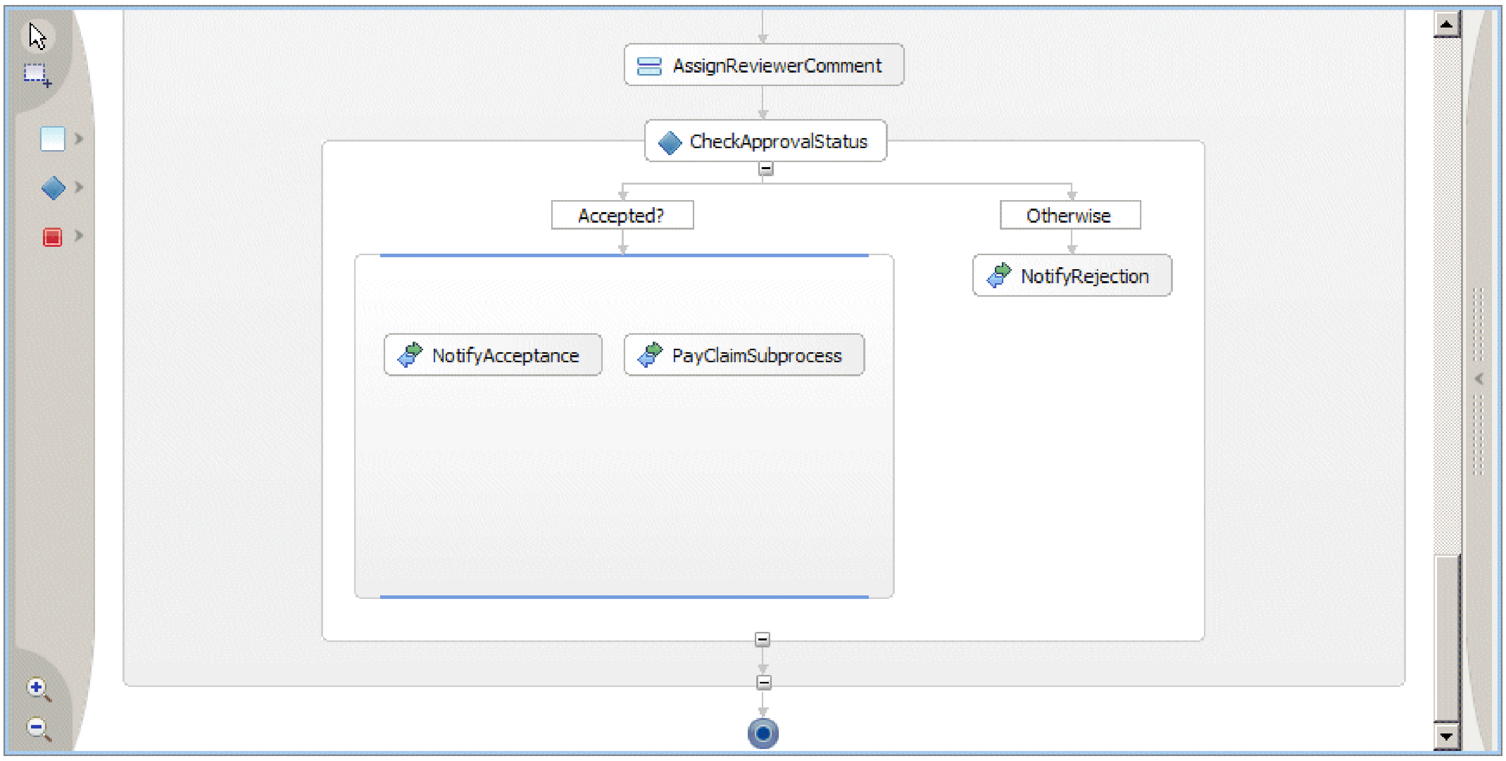


- When you click on the initial `Receive` activity, you see that this process implements the `CMProcessInterface` which is mandatory for all content-centric processes that use the Integration Toolkit.
- `GetItemPid` and `Initialize` retrieve the unique identifier of the item that is routed through the process and makes it available as a process property (see chapter 4 for details.
- `RetrieveClaimNumber`, `RetrievePolicyNumer` and the subsequent Java snippets (`SetClaimNumber` and `InitAddPolicyToFolder` do some further initialization by retrieving attribute values from DB2 Content Manager.
- `AddPolicyToFolder` invokes the **Folder Management** service to retrieve the corresponding insurance policy. Click `InitAddPolicyToFolder` and inspect the details view to see the Content Manager XPath query that identifies the appropriate policies: `/XYZ_InsPolicy[@XYZ_PolicyNum=\"" + policyNum + "\" AND @VERSIONID = latest-version(.)]`.

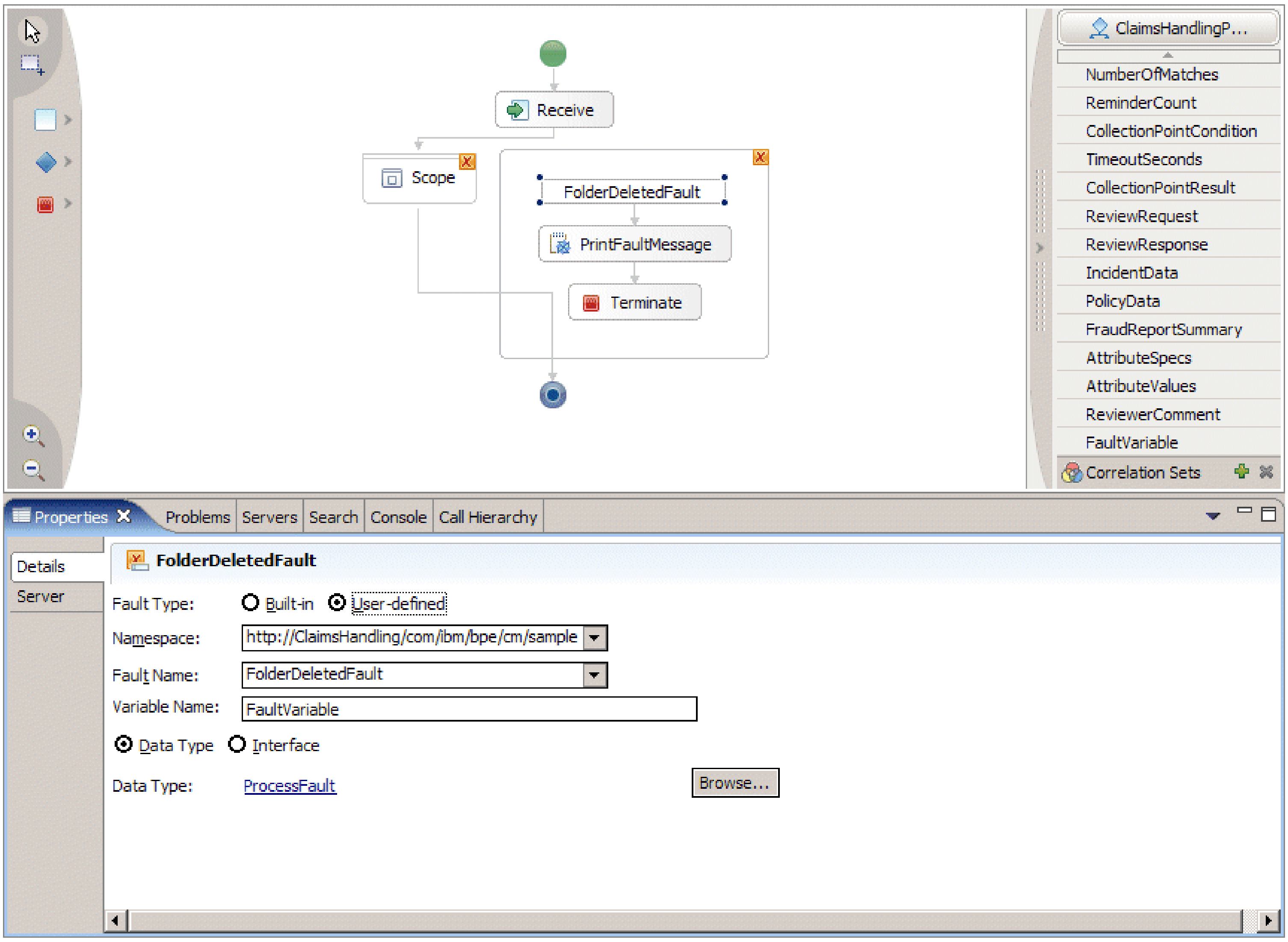
- `DisplayNumRetrievedPolicies` is a Java snippet that displays the result of the policy retrieval
- The assign activity `DefineWaitCondition` initializes the reminder count (number of loops due to collection point timeout) and the document types and quantities required to proceed with the process (collection point condition).
- Note that `DefineWaitCondition` initializes the response variable of the `ValidateAndAdjust` activity since the first activity inside the loop (`SetValidateAndAdjustRequest`) copies the values from the response to the request variable to ensure changes which tha adjuster may have made to the collection point condition are retained throughout the loop.
- `WhileDocumentsAreNotReady` is a structured activity that executes a loop while its entry condition is true. Click its box and check out the **Details** to see that the entry condition is based on the reminder count. Setting it to a negative value forces the loop to terminate.
- The first activity inside the loop (`SetValidateAndAdjustRequest`) is an assignment activity that prepares the input message for the human activity that follows (`ValidateAndAdjust`) by initializing the number of retrieved contracts to the results of the number of matches returned from the `AddPolicyToFolder` service and the number of iterations to the reminder count.
- The human activity `ValidateAndAdjust` corresponds to a **Participating Task** `ClaimsHandlingProcessTask3`. Clicking the link to this task in the **Details** view shows the staff settings. Click **Potential Owner** in the **Receiver settings** section to see that its verb is **Group Members** with a group name of `AdjustersGroup`. Close the task definition to return to the process.
- `InitCollectionPointService` and `SetTimeoutValue` define the parameters needed for the collection point service such as the timeout in seconds and the item types and respective quantities to wait for. The timeout value is set in a separate Java snippet (instead of an assign activity) since it involves a type change from String to Integer. The Quick Start Client that serves as user interface for the participating tasks relies on the convention that all tasks return data in String format which makes this conversion necessary.
- `CollectionPointService` invokes an asynchronous mechanism that lets the process wait until the folder that is routed through the process satisfies the conditions defined by its input parameters. See section 4.5 for details.
- The collection point service is followed by a structured activity **CheckCollectionPointResult** that consists of three case activities corresponding to a certain outcome of the collection point and an otherwise activity. If the condition corresponding to a case evaluates to true, the activities within its scope are executed. If none of them evaluates to true, the process continues with the 'otherwise branch'.
- The three case statements evaluate the result value that is passed back from the collection point service. Note that their Java implementation makes use of the constants defined in the `CMBPCConstants` interface of the `CMBPCIntegrationLibrary`.
- `PrepareLoopExit` and `Remind` are Java snippets that print a status message to the console output and leave (condition met) or re-enter (timeout) the loop.
- Both `FolderDeleted?` and `Otherwise` signal undesirable results and are consequently followed by activities that initialize an error message and trigger and throw an exception. Note that the process is enclosed by a scope with an exception handler that ensures that any exception triggered within the process is properly caught and displayed on the console.



- The **Review** section of the process starts with an assign activity `PrepareAttributeLookup` that initializes the fraud History value and specifies the attribute values that are retrieved for subsequent use in the remainder of the process: the first and last name of the client who submitted the claim, the date of incident and the adjusters estimate of the claim amount.
- `LookupContentAttributes` retrieves the corresponding values and `AssignValues` assigns them to input messages for the human tasks that implement the review.
- `SendNotificationLetter` is an operation of the `CommunicationService` that sends a notification back to the client (implemented as a Java method that displays a mail skeleton on the server log).
- The structured activity `CheckClaimAmount` contains a case activity that evaluates the adjuster's estimate of the claim amount. If it is small, it continues with the `ReviewSmallClaim` activity. Otherwise it invokes the `ExternalFraudHistoryService` and then the `ReviewLargeClaim` activity.
- Clicking on the activities that represent participating tasks and opening the task definition in the editor shows that the corresponding potential owner groups are `UnderwriterAssistantsGroup` and `UnderwritersGroup` respectively.
- The `PrepareFraudHistorySummary` activity in the 'otherwise branch' assigns the result of the `ExternalFraudHistoryService` to the input message of the `ReviewLargeClaim` activity.



- The first activity of the **Completion** section (`AssignReviewComment`) pulls the reviewer's comment from the output message of the review activity (both are of the same type) and stores it in a variable `ReviewerComment` that will be given as input parameter to the communication service for sending the review notification to the client.
- The case activity `Accepted?` evaluates the `decision` property of the reviewer's output message. If true, it proceeds to the parallel section that contains two activities: `NotifyAcceptance`, an operation of the communication service that notifies the client about the acceptance of the case and `PayClaimSubprocess` that performs the corresponding financial transaction.
- If the case has not been accepted, the 'otherwise path' is taken which sends a notification to the client that the case has been rejected (`NotifyRejection`).
- Note that the mail skeleton displayed in the servers output console contains a section between angle brackets that contains the reviewer's comment. This is meant as input to a 'communication department' which would replace this comment with the proper wording, double check the mail text and then send it out to the client.
- At the end of the process you might expect some archival activity that triggers proper retention management for the case. A simple version of this could be implemented based on the `setACL` service of the Integration Toolkit. However, changing the ACL of items may involve additional administrative steps to ensure proper deletion when cleaning up data associated with the sample. To keep management of the sample and its data simple we have not incorporated such a step. See section 6.3 for details on how to clean up data used by the sample process.



FolderDeletedFault

Fault Type: Built-in [✓ User-defined]

Namespace: http://ClaimsHandling/com/ibm/bpe/cm/sample

Fault Name: FolderDeletedFault

Variable Name: FaultVariable

[✓ Data Type] Interface

Data Type: ProcessFault

There are a number of situations in which a step of the process might fail. The sample process illustrates the use of a fault handler by encapsulating all activities following the initial receive activity within a scope and attaching a fault handler to this scope. This fault handler catches faults of the type `FolderDeletedFault` that occur if the folder on which an active collection point condition is registered is deleted. See the throw activity `ThrowFolderDeletedFault` in the **Data Validation** segment of the process for a cause of this fault. The fault is thrown as a consequence of the `COLLECTIONPOINT_ITEM_DELETED` result code of the collection point service. The `Initialize` activity at the beginning of the process shows how faults can be thrown from within a Java snippet. Information about the fault is communicated with the variable `FaultVariable` of type `ProcessFault`. In our example `ProcessFault` only specifies a `messageText`. It may be extended to provide more details about the fault. By adding further fault handlers to the scope other types of faults can be managed accordingly.

4. Developing a Custom DB2 Content Manager / WebSphere Process Server 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 the Integration Quick Start workspace is organized

The following sections assume a basic familiarity with *DB2 Content Manager Enterprise Edition V8.3*, *WebSphere Integration Developer V6*, and *WebSphere Process Server V6*. 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 chapter 7 at the end of this document. From the point of view of business process creation we recommend the Business Integration samples that can be found in the **Samples Gallery** of the local *WebSphere Integration Developer* Information center which can be opened by clicking **Help > Samples Gallery**. We recommend working through at least one of the 'Build it yourself' style samples such as the **Loan application** sample in the **Application Samples > Business Integration** section and taking the time to look up new concepts in the redbooks or documentation.

From a resource perspective, a workspace of a custom solution that integrates DB2 Content Manager with WebSphere Process Server typically consists of the following three enterprise applications each consisting of a number of modules:

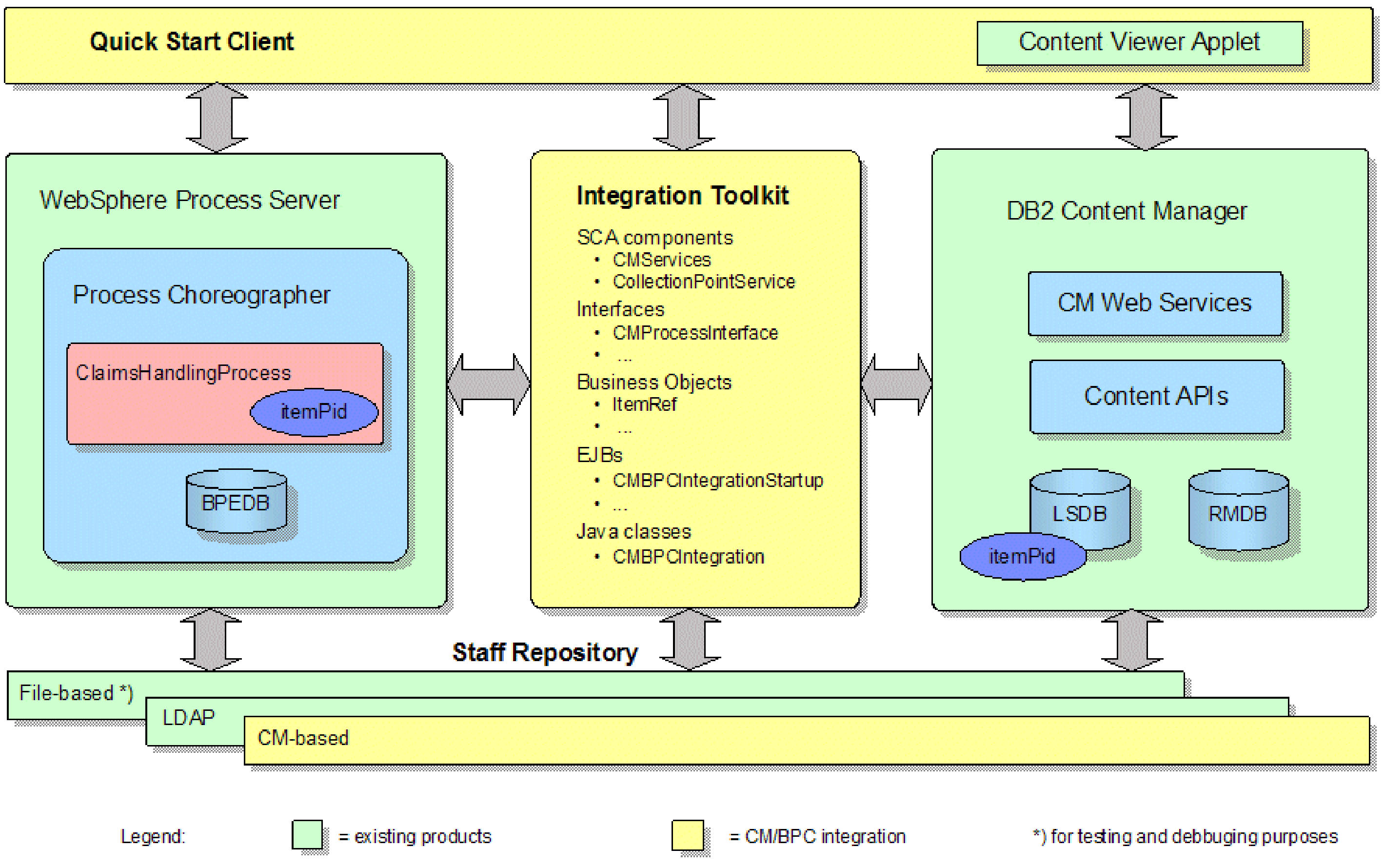
- The application hosting the business process (`ClaimsHandling` in the sample).
- The application `CMBPCIntegrationApp` hosting the modules provided by the Integration Toolkit: `CMBPCIntegrationLibrary`, `CMBPCIntegration`, `CMBPCIntegrationBeans`, `CMBPCIntegrationBeansClient`, `CMConnectionManagement`.
- The application `QuickStartClientApp` hosting the Web module `QuickStartClient`.

A solution that uses a custom Web-based (that may be built based on the JSF components of the BPC explorer) or Portal-based client only needs to deploy the first two enterprise applications while the Quick Start Client builts on top of the functions provided by the Integration Toolkit.

The following sections describe the individual elements of the Integration Toolkit and Quick Start Client that can be used to build on custom integration solution in more detail.

4.2 The Integration Toolkit

The Integration Toolkit is a set of artifacts (SCA components, Interfaces, Business Objects, EJBs, and Java classes) that support the creation of content-centric processes. See the figure below for a schematic representation of the Integration Quick Start architecture:



The yellow and red blocks mark the elements of the Integration Quick Start: the Claims Handling sample process, the Quick Start Client, and the Integration Toolkit. The CM-based user registry is a special case as it may be installed and used independently from the other elements of the Integration Quick Start. See section 4.9 for details on the CM-based user registry.

The Integration Toolkit provides building blocks for functions that are typically found in business processes that work with content. They are especially important for content-centric business processes but can also be found in processes that involve content but have a different main focus. This document refers to this set of common functions as content / workflow interaction patterns.

The Integration Toolkit offers building blocks for the following interaction patterns:

- *Content Event Handling* ⇒ Item creation or deletion events in DB2 Content Manager trigger corresponding actions in WebSphere Process Server.
- *Collection Point* ⇒ A step in the process where a process instance waits 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 Server and associated content attribute values from DB2 Content Manager. Typically used to create worklists for client users.
- *Common Staff Repository* ⇒ Map DB2 Content Manager user and group definitions to the Process Server staff resolution facility based on a WebSphere® custom user registry.

These building blocks are available in the form of SCA (Service Component Architecture) components, Enterprise Java Beans, and Java classes. The SCA components are the main vehicle for incorporating the functions offered by the Integration Toolkit into business processes. They are supported by interface definitions and business objects that define how the business process interacts with the SCA components. The EJBs provide the infrastructure needed to implement the *Content Event Handling* and *Collection Point*, and *Combined Query* patterns. The rest of this section provides some background about the available EJBs and their role within the Integration Toolkit.

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 (`CMBPCIntegrationBeans\com\ibm\bpe\cm\CMBPCIntegrationStartupBean`) to start and stop the asynchronous bean described below. Note that the startup bean starts when the enterprise application is started. It terminates when the enterprise application is stopped.

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 static methods in the `CMBPCIntegration` class which perform the following operations:

- Check for satisfied collection point criteria
- Process DB2 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 or server shutdown.

The **Stateless Session Bean** of the Quick Start Integration (`CMBPCIntegrationBeans\com\ibm\bpe\cm\CMBPCIntegrationBean`) provides a remote interface for the **Combined Query** function described in 4.8.

4.3 Creating a process based the Integration Toolkit

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

- It is based on the `CMProcessInterface` provided by the `CMBPCIntegrationLibrary`.
- It has a single receive activity.
- This receive activity is followed by an activity that invokes the `createItemRef` operation of the `CMServices` component
- This invoke activity is followed by a Java snippet that initializes custom properties that are needed for the Combined Query function of the Integration Toolkit.

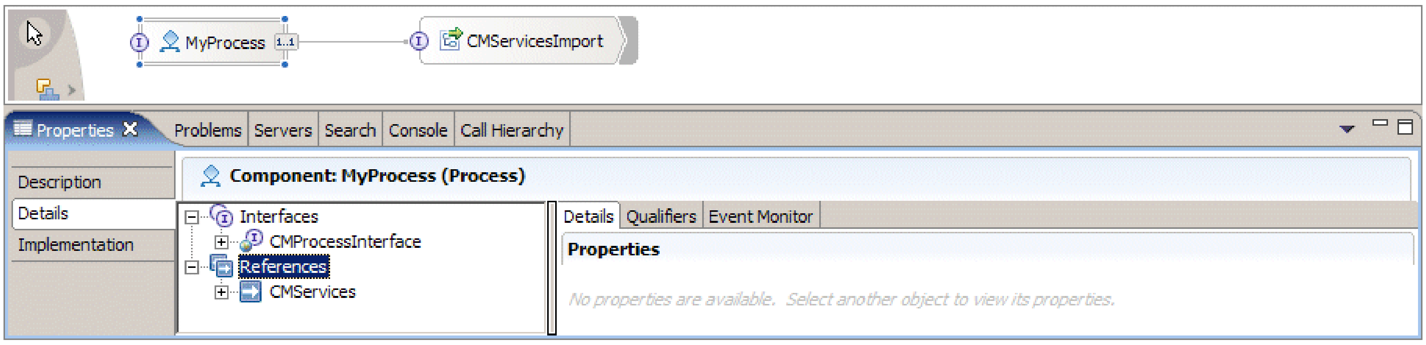
Details can be found in the sample process or in the step-by-step instructions below.

To create a new content-centric process proceed as follows:

1. Start WebSphere Integration Developer and switch to the Business Integration view.
2. In the menu click `New > Project > Business Integration > Module` and enter a name for the Module such as `MyModule` (the default location should be OK).
3. In the project tree right-click `MyModule` and select **Open Dependency Editor**. In the **Libraries** section of the dependency editor click **Add** and select `CMBPCIntegrationLibrary`. Press CTRL-S to store the change and close the dependency editor.
4. Double-click the assembly diagram of your new module to open the assembly editor.
5. Drag `CMProcessInterface` from `CMBPCIntegrationLibrary > Interfaces` and drop it into the assembly editor and select **Component with no implementation type**.
6. Select the new component and replace its default name `Component1` with a more useful name in the properties view such as `MyProcess`.
7. Right-click the component `MyProcess` and select **Generate Implementation > Process**. On the **Generate Implementation** dialogue you can define a folder for the process such as `com.ibm.bpe.cm.myprocess`.
8. When you click **OK** a new process definition `MyProcess` is created based on the interface partner `MyProcessComponent` and is displayed with the process editor.
9. In the **Variables** section to the right of the business process editor create five variables with the data types shown below:

Variable name	**Data Type**
`ItemId`	`string`
`ComponentId`	`string`
`VersionNo`	`int`
`ItemPid`	`string`
`ItemRef`	`ItemRef`

10. Click the **Receive** activity and select the **Details** tab in the **Properties** view. Check **Use Data Type Variables** and assign the three variables `ItemId`, `ComponentId`, and `VersionNo` to the corresponding input parameters of the `startProcess` operation. You may now delete `InputVariable` from the list of variables.
11. Drag `CMServicesInterface` from `CMBPCIntegrationLibrary > Interfaces` and drop it into the **Reference Partners** section at the right-hand side of the business process editor.
12. Drag the symbol of an **Invoke** activity from the graphical menu of the business process editor and drop it right below the Receive activity (within the Sequence block).
13. Click the new Invoke activity. In the **Description** tab of the **Properties** view replace the default name `Invoke` with `CreateItemRef` and in the `Details` tab click **Browse** and select the `CMServicesInterface` partner. Select the operation `createItemRef` from the list and assign the input variables (`ItemId`, `ComponentId`, `VersionNo`) to the corresponding input parameters and the output variable (`ItemRef`) to the output parameter.
14. Press CTRL-S to store the project definition.
15. Switch to the assembly editor. Note that the referenced partners have now changed so the assembly diagram needs to be updated. Right-click `MyProcess` and select **Delete**. Press CTRL-S to update the assembly diagram and wait for the progress information to signal completion of the build. Drag `MyProcess` from `MyModule > Business Logic > Processes` and drop it into the assembly diagram. Note that it now contains a small icon **1:1** on the right hand side that represents its partner link to `CMServices`.
16. Drag `CMServicesInterface` from `CMBPCIntegrationLibrary > Interfaces` and drop it into the assembly editor right of `MyProcess`. Select **Import with no binding** on the **Component Creation** dialogue.
17. Click the new component and replace its default name `Import1` in the **Description** tab of the **Properties** view with `CMServicesImport`.
18. Right-click `CMServicesImport` and select **Generate Binding > SCA Binding**.
19. Select the **Binding** tab in the **Properties** view and click **Browse**. Select `CMServicesExport` on the **SCA Export Selection** dialogue and click **OK**.
20. In the assembly editor right-click `MyProcess` and select **Wire to existing**. Note that this connects the process component `MyModule` with its referenced partner `CMServicesImport`.
21. Press CTRL-S to save the changes to the assembly diagram. It should now look as shown below:



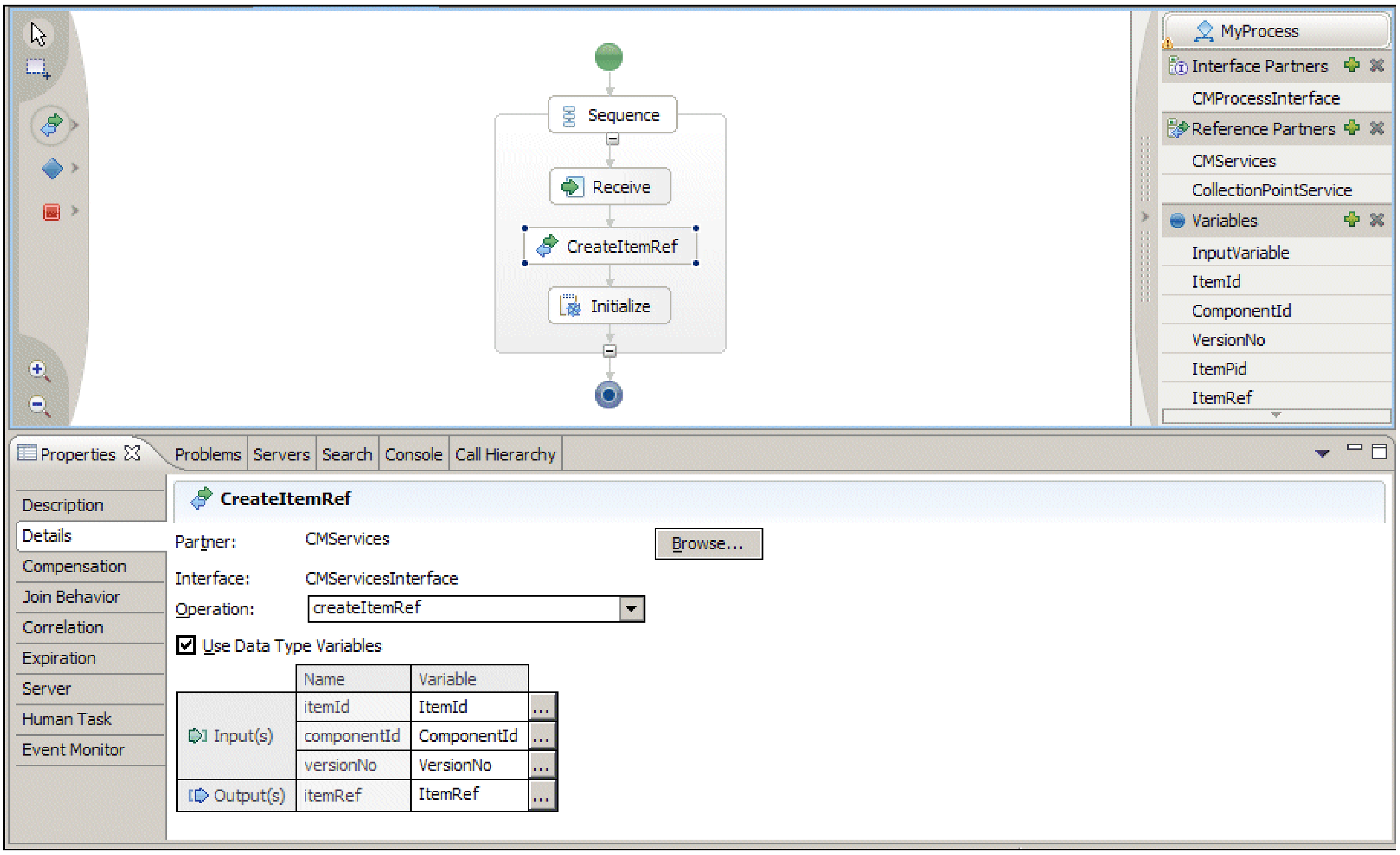
No properties are available. Select another object to view its properties.

22. Switch to the business process editor, click the background of the process definition and select the **Java Imports** tab in the **Properties** view. Add `import com.ibm.bpe.cm.CMBPCConstants;` to the (so far empty) list of Java imports.
23. Drag the symbol of a **Java Snippet** from the graphical menu on the left and drop it right below the `CreateItemRef` Invoke activity.
24. In the **Description** tab of the **Properties** view replace the default name `Snippet` with `Initialize` and switch to the **Details** tab.
25. Select **Java**. On the confirmation dialogue click **Yes** and paste the following code fragment into the source code editor area of the **Details** view.

```
ItemPid = ItemRef.getString(CMBPCConstants.BOATT_ITEM_REF_PIDSTRING);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_ITEMPID, ItemPid);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_ITEMID, ItemId);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_COMPONENTID,
ComponentId);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_VERSIONNUMBER,
VersionNo.toString());
```

26. Press CTRL-S to store the updated process definition.
27. Add further activities to the process as needed and proceed as described in 6.6 to deploy and run your new process.
28. It is recommended to have a scope at the top level of the process so dedicated treatment can be established for errors that may show up during the process. The Claims Handling sample process shows an example how this can be done.

The `CreateItemRef` activity invokes the `createItemRef` operation of the `CMServices` component. Its implementation creates a `DKItemPid` object from the supplied `ItemId`, `ComponentId`, and `VersionNo` parameters and runs a Content Manager query that returns the missing information such as the server name and item type from the Content Manager Library server.



CreateItemRef

Partner: CMServices

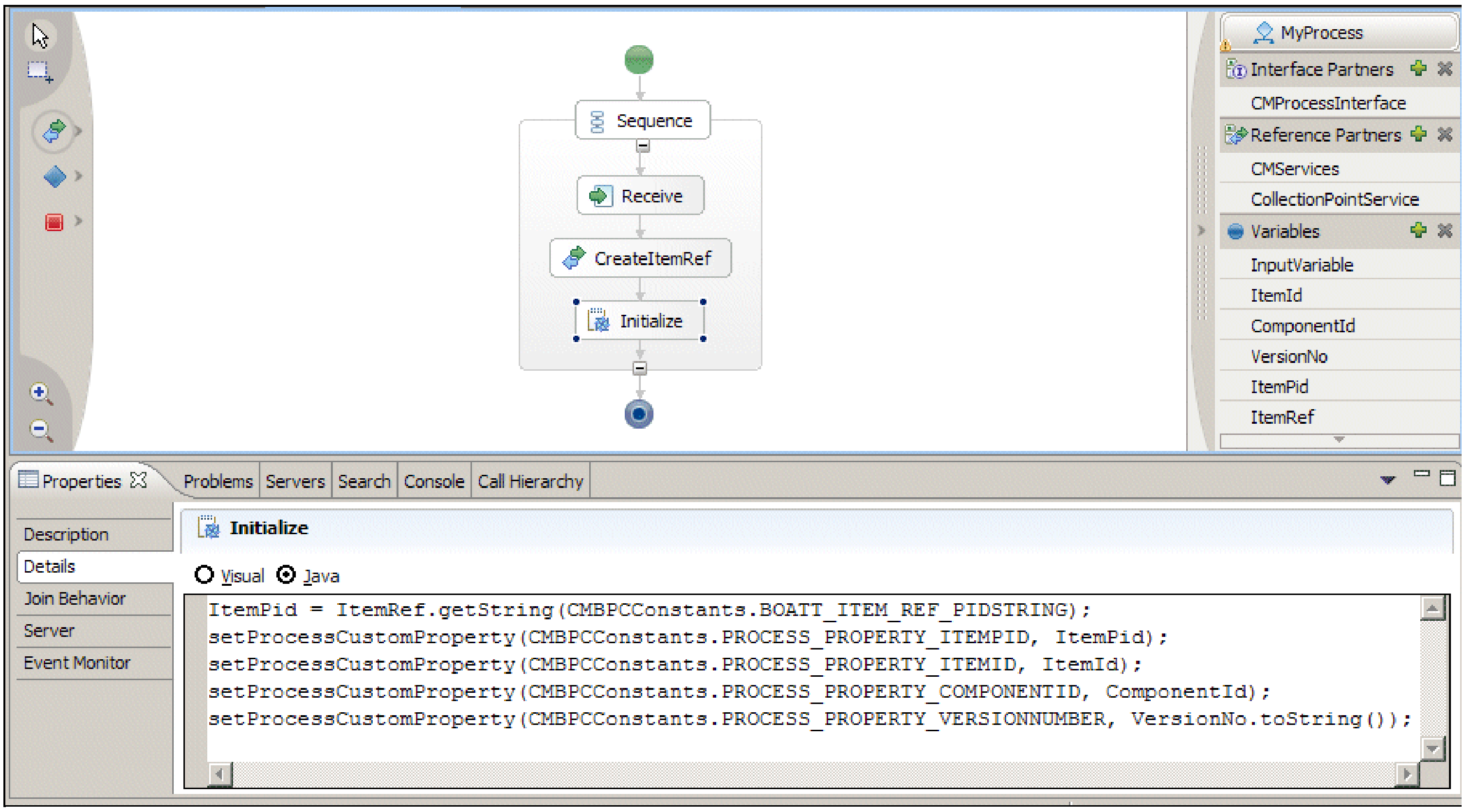
Interface: CMServicesInterface

Operation: createItemRef

[✓] Use Data Type Variables

	Name	Variable
Input(s)	itemId	ItemId
	componentId	ComponentId
	versionNo	VersionNo
Output(s)	itemRef	ItemRef

The Java snippet `Initialize` copies the values of the four variables `ItemId`, `ComponentId`, `VersionNo`, and the ItemPid stored in the variable `ItemRef` into custom properties of the process where it can be used by Integration Toolkit functionality such as Combined Queries or Collection Points.



Initialize

Visual [✓ Java]

```
ItemPid = ItemRef.getString(CMBPCConstants.BOATT_ITEM_REF_PIDSTRING);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_ITEMPID, ItemPid);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_ITEMID, ItemId);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_COMPONENTID, ComponentId);
setProcessCustomProperty(CMBPCConstants.PROCESS_PROPERTY_VERSIONNUMBER, VersionNo.toString());
```

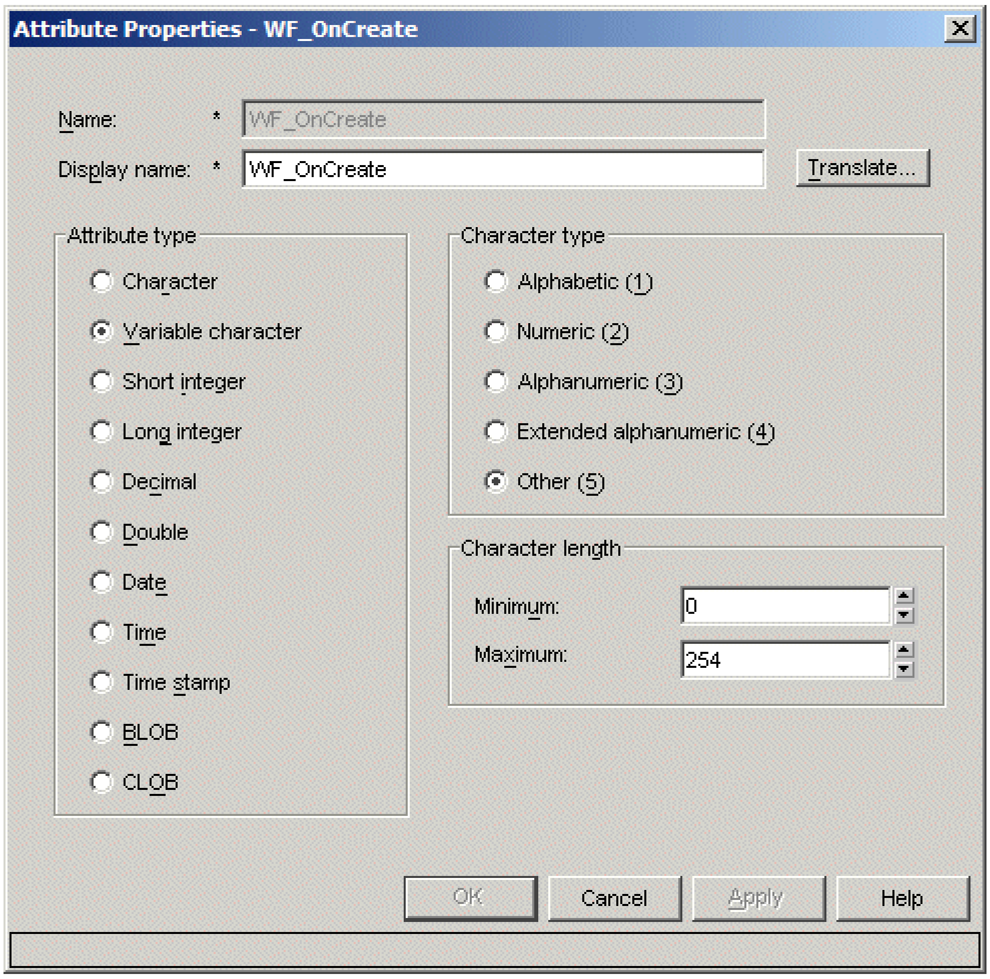
The following sections describe the individual building blocks for content-centric processes that are provided by the Integration Toolkit in more detail.

4.4 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 DB2 Content Manager 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

Name: * WF_OnCreate

Display name: * WF_OnCreate

Attribute type: Variable character

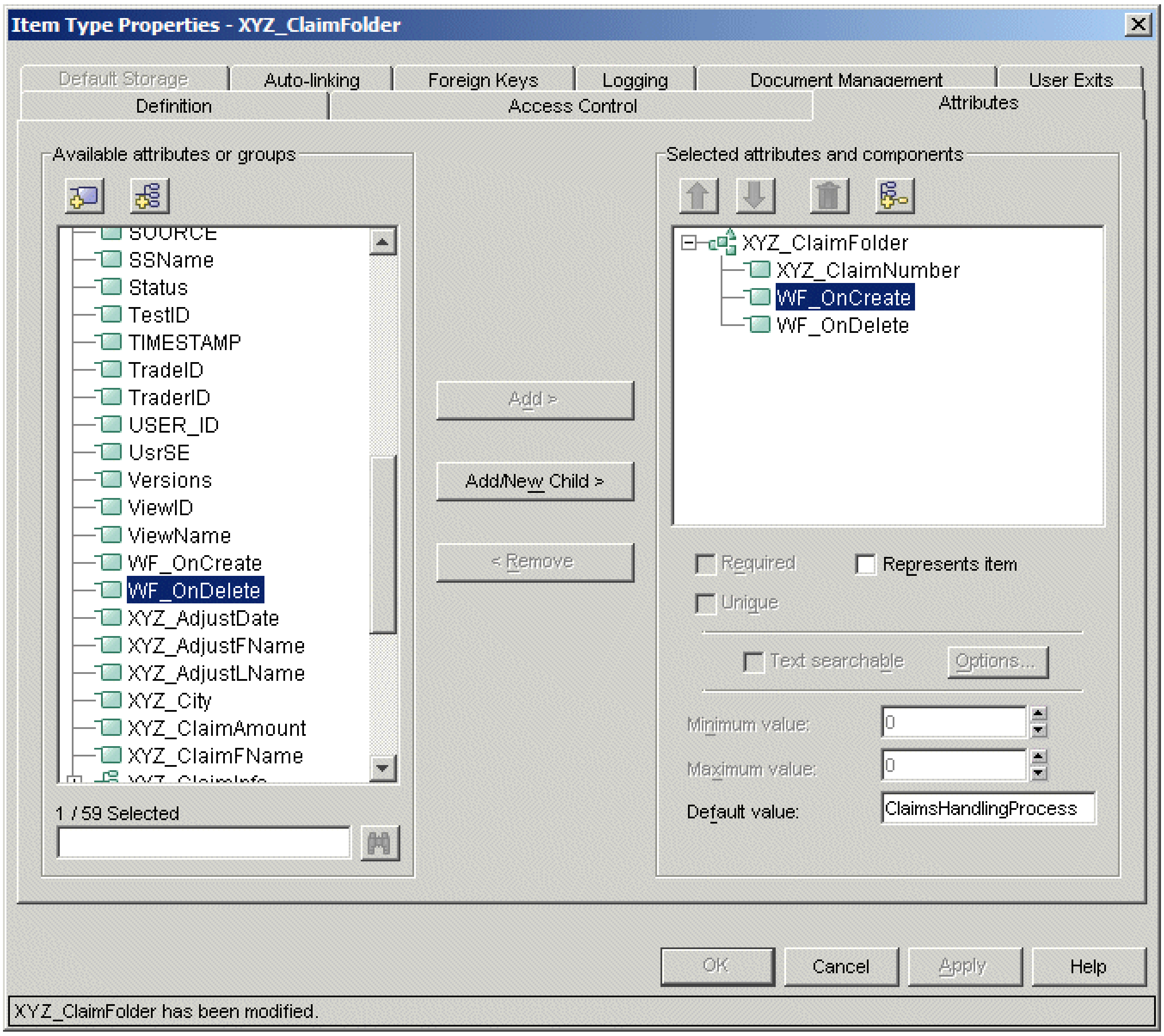
Character type: Other (5)

Character length

Minimum: 0

Maximum: 254

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.



Default value: ClaimsHandlingProcess

XYZ_ClaimFolder has been modified.

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 process `ClaimsHandlingProcess` is started by submitting the item's `itemId`, `componentId`, `versionNumber` as input message to the process which needs to implement the appropriate interface of type `CMProcessInterface`. 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 process `ClaimsHandlingProcess` that carry this item.

With this synchronization mechanism, 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 in synch.

How Content Event Handling is implemented

Item type definitions that have a `WF_OnCreate` or `WF_OnDelete` attribute are registered in a table BPECONTENTENVENTS 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 utility class `CMBPCIntegration\com\ibm\bpe\cm\util\ContentEventHandling` is called regularly from the asynchronous bean (`CMBPCIntegration\com\ibm\bpe\cm\util\AsyncBeanWork`) to see if a new process instance should be created or if process instances should be deleted. It scans the BPECONTENTEVENTS table 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 by the startup bean (CMBPCIntegrationBeans\com\ibm\bpe\cm\util\CMBPCIntegrationStartupBean) if it does not exist yet.

The BPECONTENTEVENTS table has the following structure:

```
CREATETS         TIMESTAMP
ITEMID           CHAR(26)
COMPONENTID      CHAR(18)
VERSIONID        SMALLINT
REQUEST          SMALLINT
PROCESSNAME      VARCHAR(254)
```

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 or deleted. Rows are written to the BPECONTENTEVENTS table using database triggers. These triggers are automatically created by the Asynchronous Bean for DB2 Content Manager item types that contain the custom attributes WF_OnCreate or WF_OnDelete. The trigger is created on the underlying DB2 Content Manager Component Table (a table of the form ICMUTnnnnnsss where nnnnn and sss are sequences of numbers representing the component type ID and the segment ID respectively) by the startup bean. The startup bean deletes all existing triggers and replaces them with triggers that correspond to the current state of the library server database. Note that when adding or removing WF_OnCreate or WF_OnDelete attributes to/from item types, the server needs to be re-started (or the CMBPCIntegrationApp re-deployed) to ensure the Content Event Handling mechanism can pick up the change.

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 DB2 Content Manager transaction from the subsequent Check Content Event transaction so that the performance impact to the DB2 Content Manager transaction is minimal. The BPECONTENTEVENTS table is created by the startup bean if it does not yet exist.

In addition to this 'standard usage', 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:

- 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.

4.5 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 an invoke activity that invokes the collectionPoint operation on the CMServices component.

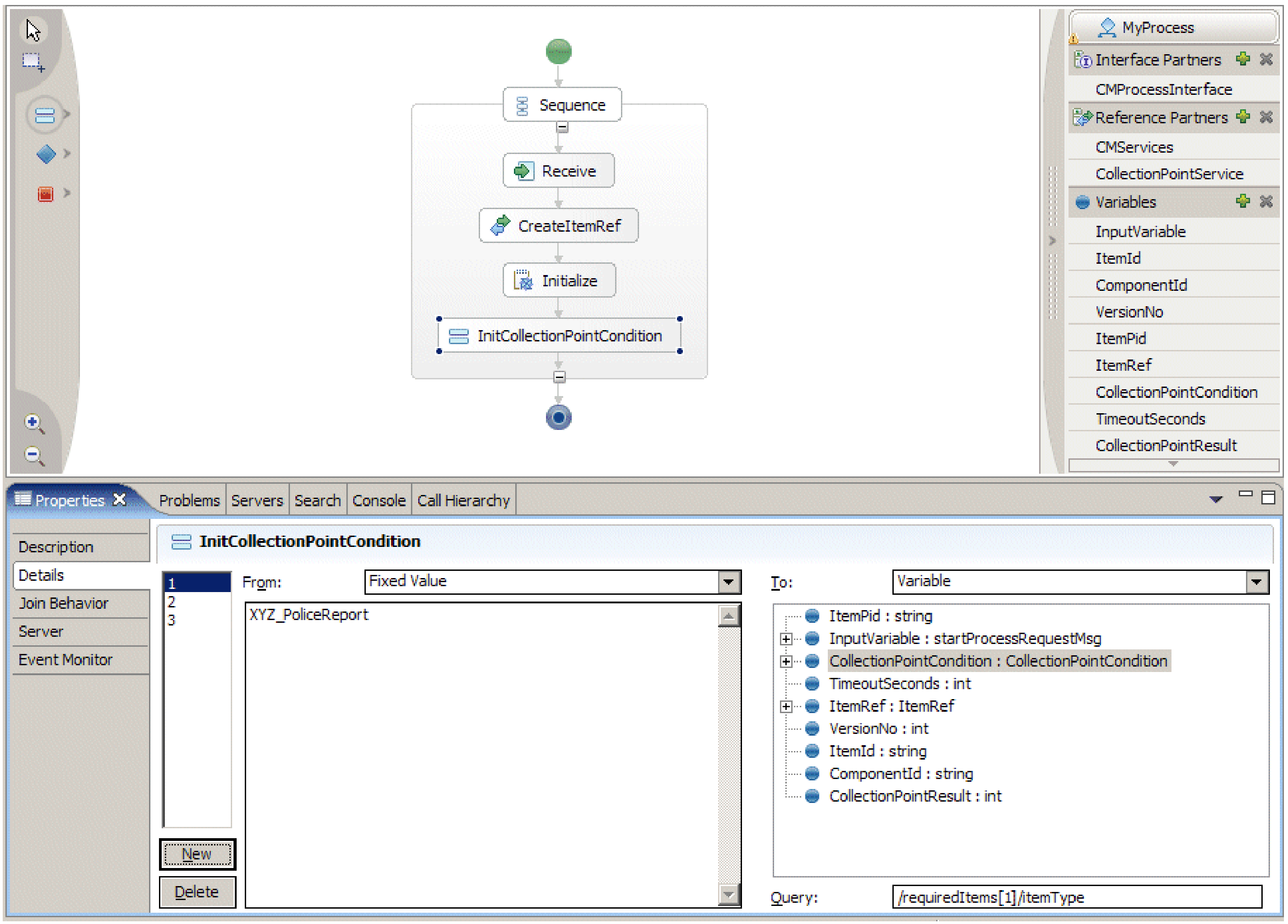
To add a collection point service invocation to a content-centric process proceed as follows:

1. In the **Variables** section to the right of the business process editor create three variables with the data types shown below:

Variable name	Data Type
CollectionPointCondition	CollectionPointCondition
TimeoutSeconds	int

```
CollectionPointResult        int
```

2. Drag `CollectionPointService` from `CMBPCIntegrationLibrary > Interfaces` and drop it into the **Reference Partners** section at the right-hand side of the business process editor.
3. Drag the symbol of an **Assign** activity from the graphical menu of the business process editor and drop it to the position where the collection point service should be inserted.
4. Click the new Assign activity. In the **Description** tab of the **Properties** view replace the default name `Assign` with `InitCollectionPointCondition` and in the `Details` tab add three assignments to this activity as shown below (for illustration purposes we are using fixed values based on the sample data. Note that `requiredItems` is an array. To assign a value to elements of this array select **Variable** in the target section and click `CollectionPointCondition... > requiredItems... > itemType`. Note that this only displays the scalar variant of the query: `/requiredItems/itemType`. You need to manually enter the index value of the array into the query field so that it displays `/requiredItems[1]/itemType`. Note that array positions start with 1.



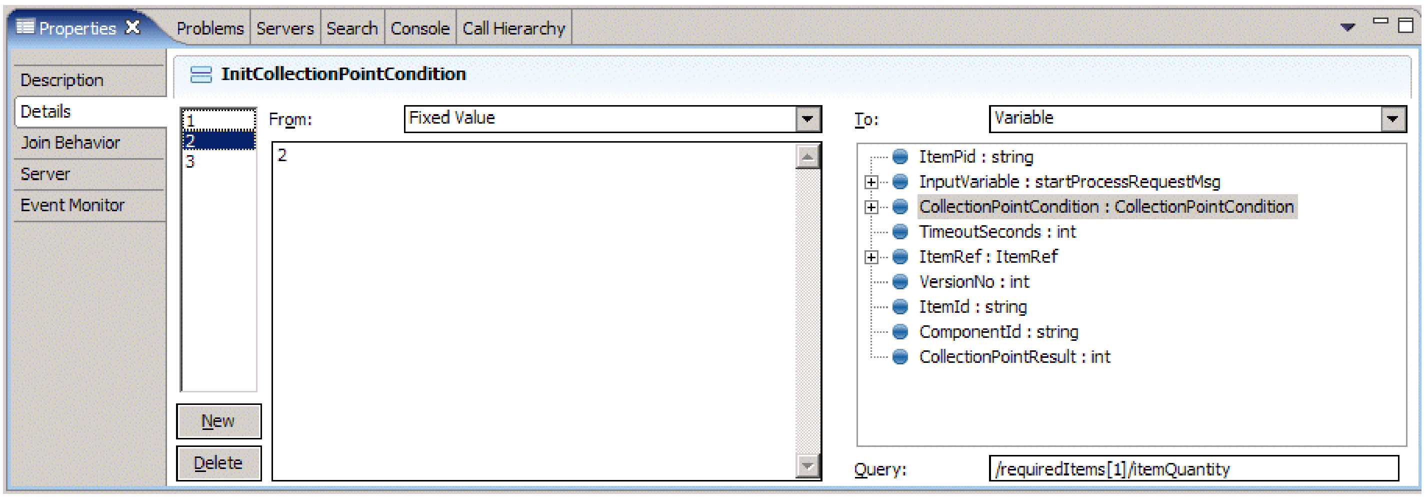
InitCollectionPointCondition

From: Fixed Value

XYZ_PoliceReport

To: Variable

Query: /requiredItems[1]/itemType



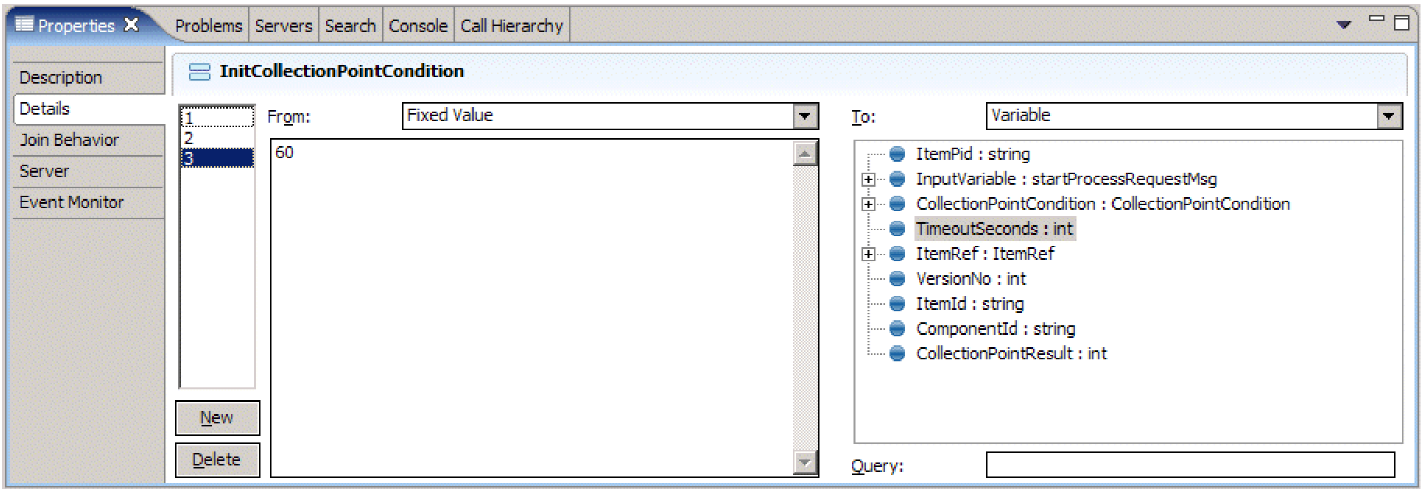
InitCollectionPointCondition

From: Fixed Value

2

To: Variable

Query: /requiredItems[1]/itemQuantity



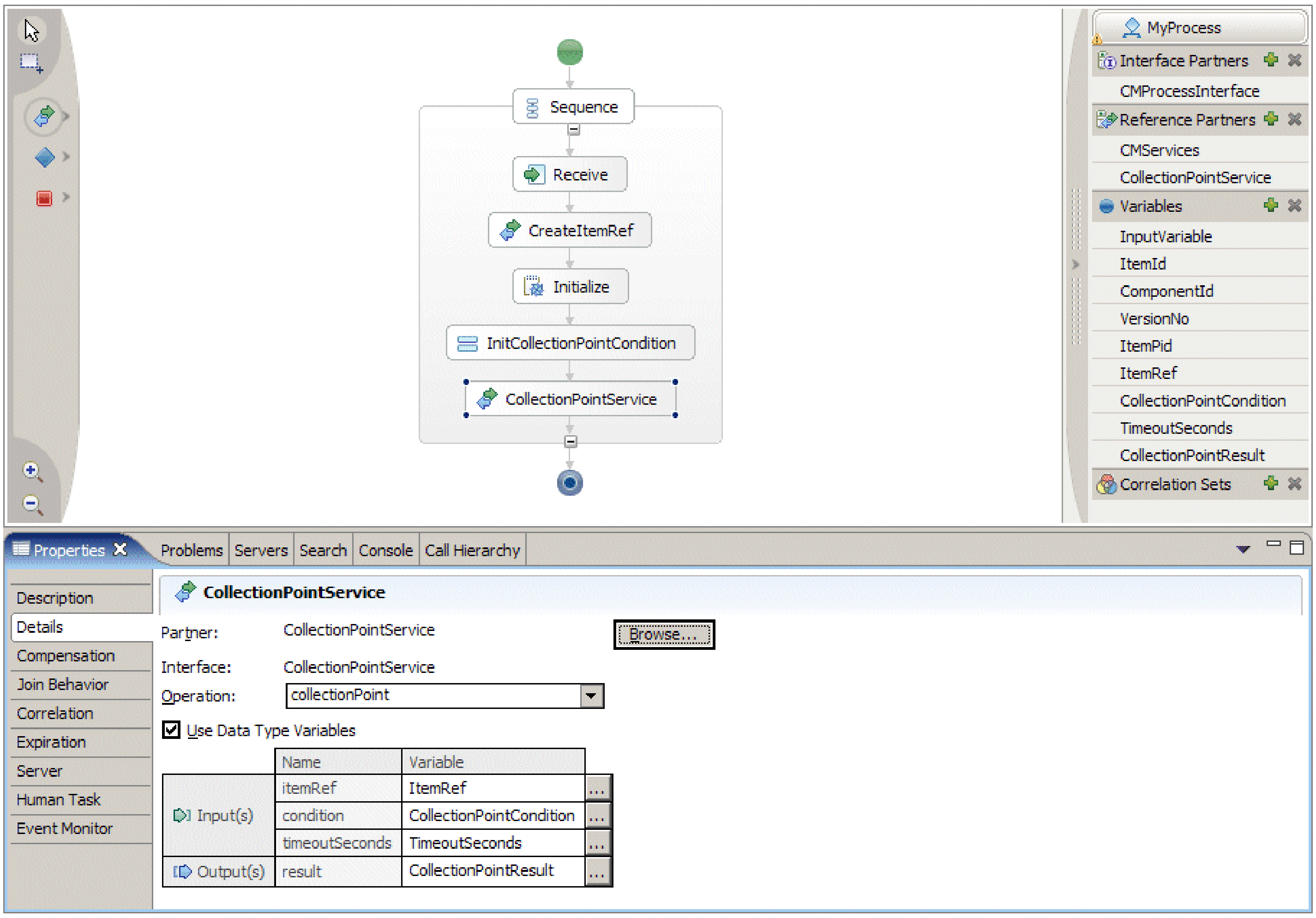
InitCollectionPointCondition

1	From: Fixed Value	To: Variable	
2	60	ItemPid : string	
3		InputVariable : startProcessRequestMsg	

Variable tree: ItemPid : string; InputVariable : startProcessRequestMsg; CollectionPointCondition : CollectionPointCondition; TimeoutSeconds : int; ItemRef : ItemRef; VersionNo : int; ItemId : string; ComponentId : string; CollectionPointResult : int

Query:

5. Press CTRL-S to store the project definition.
6. Drag the symbol of an **Invoke** activity from the graphical menu of the business process editor and drop it right below the Assign activity `InitCollectionPointCondition`.
7. Click the new Invoke activity. In the **Description** tab of the **Properties** view replace the default name `Invoke` with `CollectionPointService` and in the `Details` tab click **Browse** and select the `CollectionPointService` partner. The operation is set to `collectionPoint` since this is the only operation available at this interface.
8. Assign the input variables (`ItemRef`, `CollectionPointCondition`, `TimeoutSeconds`) to the corresponding input parameters and the output variable (`CollectionPointResult`) to the output parameter.



CollectionPointService

Partner: CollectionPointService

Interface: CollectionPointService

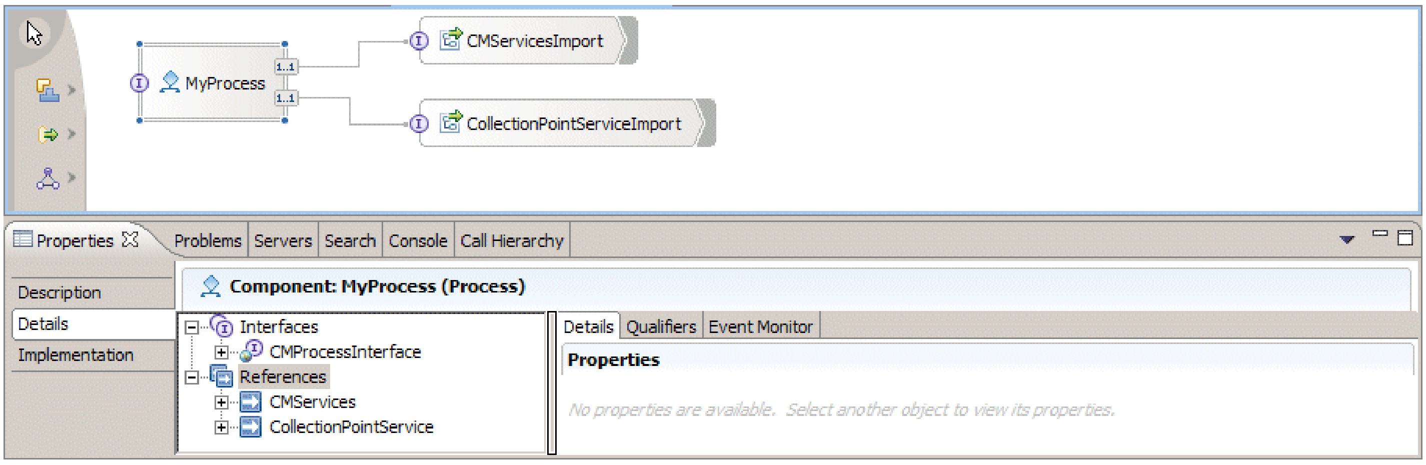
Operation: collectionPoint

☑ Use Data Type Variables

	Name	Variable
Input(s)	itemRef	ItemRef
	condition	CollectionPointCondition
	timeoutSeconds	TimeoutSeconds
Output(s)	result	CollectionPointResult

9. Press CTRL-S to store the project definition.
10. Switch to the assembly editor. Note that the referenced partners have now changed so the assembly diagram needs to be updated. Right-click `MyProcess` and select **Delete**. Press CTRL-S to update the assembly diagram and wait for the progress information to signal completion of the build. Drag `MyProcess` from `MyModule > Business Logic > Processes` and drop it into the assembly diagram. Note that it now contains a small icon **1:1** on the right hand side that represents its partner link to `CMServices`.

11. Drag `CollectionPointService` from `CMBPCIntegrationLibrary > Interfaces` and drop it into the assembly editor right of `MyProcess`. Select **Import with no binding** on the **Component Creation** dialogue.
12. Click the new component and replace its default name `Import1` in the **Description** tab of the **Properties** view with `CollectionPointServiceImport`.
13. Right-click `CollectionPointServiceImport` and select `Generate Binding > SCA Binding`.
14. Right-click `MyProcess` and select **Wire to existing**. Note that this connects the process component `MyModule` with its referenced partner `CMServicesImport`.
15. Press CTRL-S to save the changes to the assembly diagram.

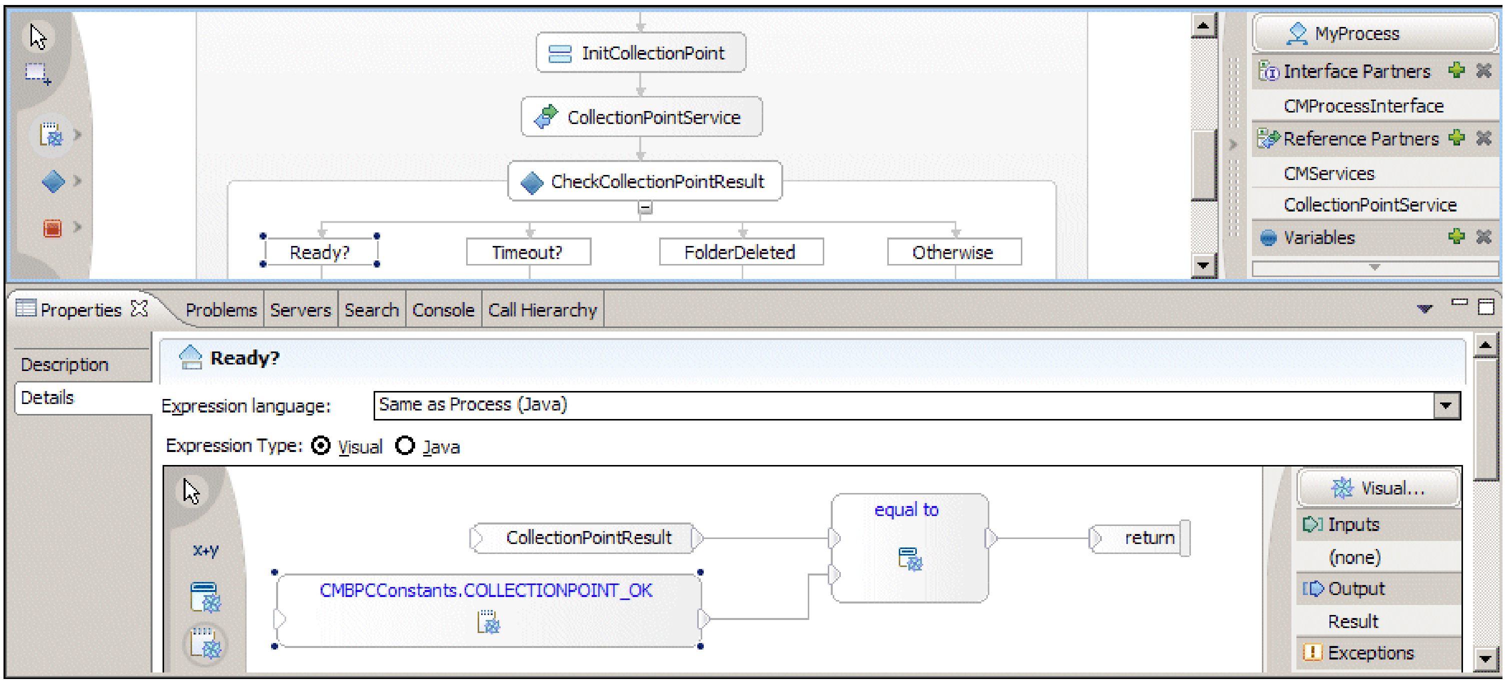


Note that a process using a collection point must be defined as long running since the availibility of documents usually depends on some external conditions that are beyond the control of the process and typically person-based. Furthermore, a timeout may be specified to ensure the process instance resumes execution in lack of the required documents. The timeout value is typically used to trigger a reminder or corrective action to ensure the process instance does not wait forever.

The `CollectionPointResult` parameter 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 seconds specified by the timeout parameter of the input message
- *Folder has been 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:



Expression language: Same as Process (Java)

Expression Type: [✓ Visual] Java

CMBPCConstants.COLLECTIONPOINT_OK

The **Assign** activity instantiates the input parameters of the collection point service. It submits the reference to the folder (`ItemRef`), and defines the collection point condition and timeout value as shown above. The case statements inside the **Choice** activity evaluate 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. Invoking the collection point service and evaluating its results is typically done inside a loop so the collection point service can be re-entered in case of a timeout as shown by the Claims Handling sample process. If the initialization also happens within the loop, collection point conditions and timeout values can be modified dynamically with each iteration to, for example, loosen the conditions in case of a timeout.

How the Collection Point Service is Implemented

Since the class implementing the collection point SCA component (`CMBPCIntegration\com\ibm\bpe\cm\sca\CollectionPointServiceImpl`) instantiates the `ServiceImplAsync` interface, the SCA infrastructure redirects calls to any of its operations (there only is one: `collectionPoint`) to the method `invokeAsynch` which has a caller ticket and service callback as parameters. `invokeAsynch` calls the `collectionPoint` method in `CMBPCIntegration\com\ibm\bpe\cm\api\CMBPCIntegration` where in- and output parameter mapping takes place. Then the condition is registered by the `addCollectionPointRequest` method in `CMBPCIntegration\com\ibm\bpe\cm\util\CollectionPoint` that writes it to a table `BPECOLLECTIONPOINTITEMTYPES` in the DB2 Content Manager library server.

The DB2 Content Manager library server database contains two tables for collection point support provided by the Integration Toolkit.

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

```
SEQID           INTEGER NOT NULL PRIMARY KEY
CREATETS        TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP
CORRELATION     BLOB(1M) NOT NULL
ITEMID          VARCHAR(26) NOT NULL
TIMEOUT         BIGINT NOT NULL
```

The `SEQID` column contains a unique sequence ID serving as foreign key for the `BPECOLLECTIONPOINTITEMTYPES` table. The `CREATETS` column contains 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 `CORRELATION` column keeps a serialized instance of the CollectionPointCallbackInfo class that stores the `serviceCallback` and `Ticket` which the `invokeAsynch` method has obtained from the SCA infrastructure. The process engine uses this information to correlate the response message with the request message so it knows which process instance should be notified if the collection point condition evaluates to true. The `ITEMID` column is the Item ID of the folder to be checked. `TIMEOUT` stores the time in milliseconds at which the collection point condition times out

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

```
SEQID             INTEGER NOT NULL
ITEMTYPENAME      VARCHAR(254)
ITEMTYPEQUANTITY  INTEGER
```

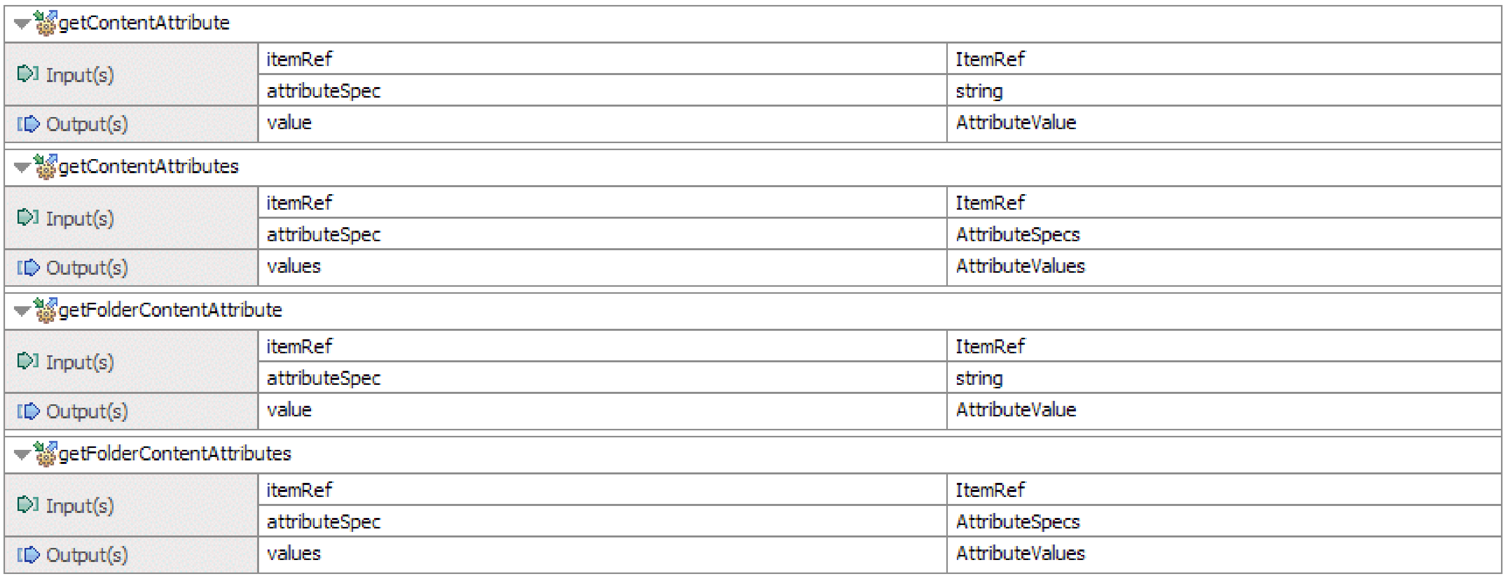
The `SEQID` references a sequence ID of `BPECOLLECTIONPOINTS` (foreign key). The `ITEMTYPENAME` column contains the name of the item type to wait for. `ITEMTYPEQUANTITY` specifies the number of items of type `ITEMTYPENAME` to wait for.

There is a one-to-many relationship between `BPECOLLECTIONPOINTS` and `BPECOLLECTIONPOINTITEMTYPES`, based on the `SEQID` key. Any of these tables is created by the `CMBPCIntegrationStartupBean` if it does not already exist.

The method `checkCollectionPoint()` that is called by the asynch bean performs a (JDBC-based) 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()` effectively joins these tables with the DB2 Content Manager Links, Items and NLS Keywords tables to identify all satisfied collection points within two requests 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 DB2 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 DB2 Content Manager item type names to numeric Item Type IDs 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).

4.6 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 `CMServices` component of the Integration Toolkit offers the following operations to extract attribute values from items or to evaluate conditions over attributes:



getContentAttribute		
Input(s)	itemRef	ItemRef
	attributeSpec	string
Output(s)	value	AttributeValue

getContentAttributes		
Input(s)	itemRef	ItemRef
	attributeSpec	AttributeSpecs
Output(s)	values	AttributeValues

getFolderContentAttribute		
Input(s)	itemRef	ItemRef
	attributeSpec	string
Output(s)	value	AttributeValue

getFolderContentAttributes		
Input(s)	itemRef	ItemRef
	attributeSpec	AttributeSpecs
Output(s)	values	AttributeValues

Each of these operations takes an item PID (`ItemRef`) as input parameter and either a single attribute specification in the form `<itemTypeName>/<attributePath>` or an `AttributeSpecs` data object that carries a list of individual attribute specifications.

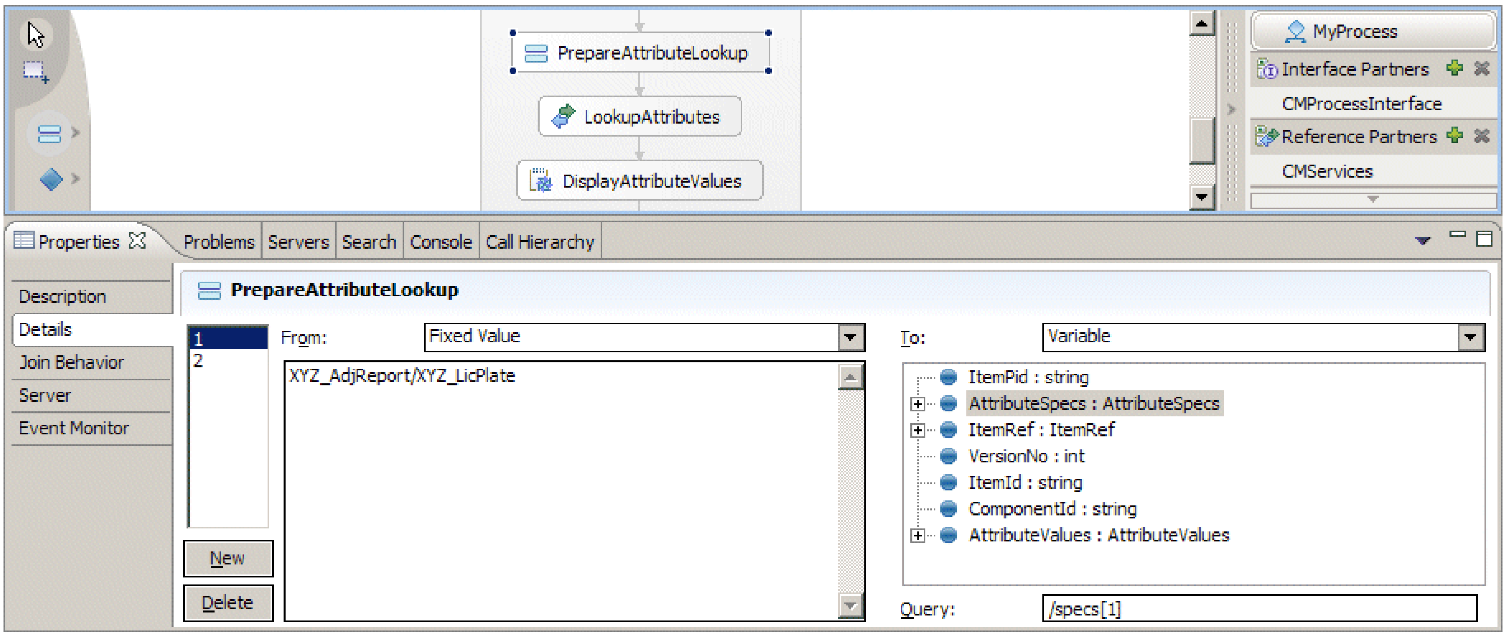
`<itemTypeName>` specifies the type of item that holds the attribute value of interest (the same attribute may be defined for many different items).
`<attributePath>` can be an atomic attribute name or a sequence of attribute names separated by '/' in case of an attribute hierarchy (as for example in `customer/address/city`).

`getContentAttribute` and `getFolderContentAttribute` are the 'single specification, single value' variants of the attribute lookup. They return a single value based on a single attribute specification and should not be used for looking up attributes of item types where more than one instance may be stored in a folder. In this case the 'multiple specification, multiple values' variants `getContentAttributes` and `getFolderContentAttributes` should be preferred. They return arrays of values where each array corresponds to an item in the folder. Note that these variants can also be used to look up multiple attribute values within a single request to the library server.

The two variants `getContentAttribute(s)` and `getFolderContentAttribute(s)` differ in that the former looks up attributes of the folder while the latter looks up attributes belonging to items within the folder. Note that only a single level of embedding is supported by the Integration Toolkit. The table below summarizes the available variants:

	attribute of folder	attribute of item in folder
single spec, single value	`getContentAttribute`	`getFolderContentAttribute`
multiple spec, multiple value	`getContentAttributes`	`getFolderContentAttributes`

The following figure shows the typical use of a multiple attribute value lookup that consists of a preparational step that initializes the attribute specifications, the lookup service invocation, and an evaluation step that (in this example) prints the attribute values to the server's console.



PrepareAttributeLookup

From: Fixed Value

To: Variable

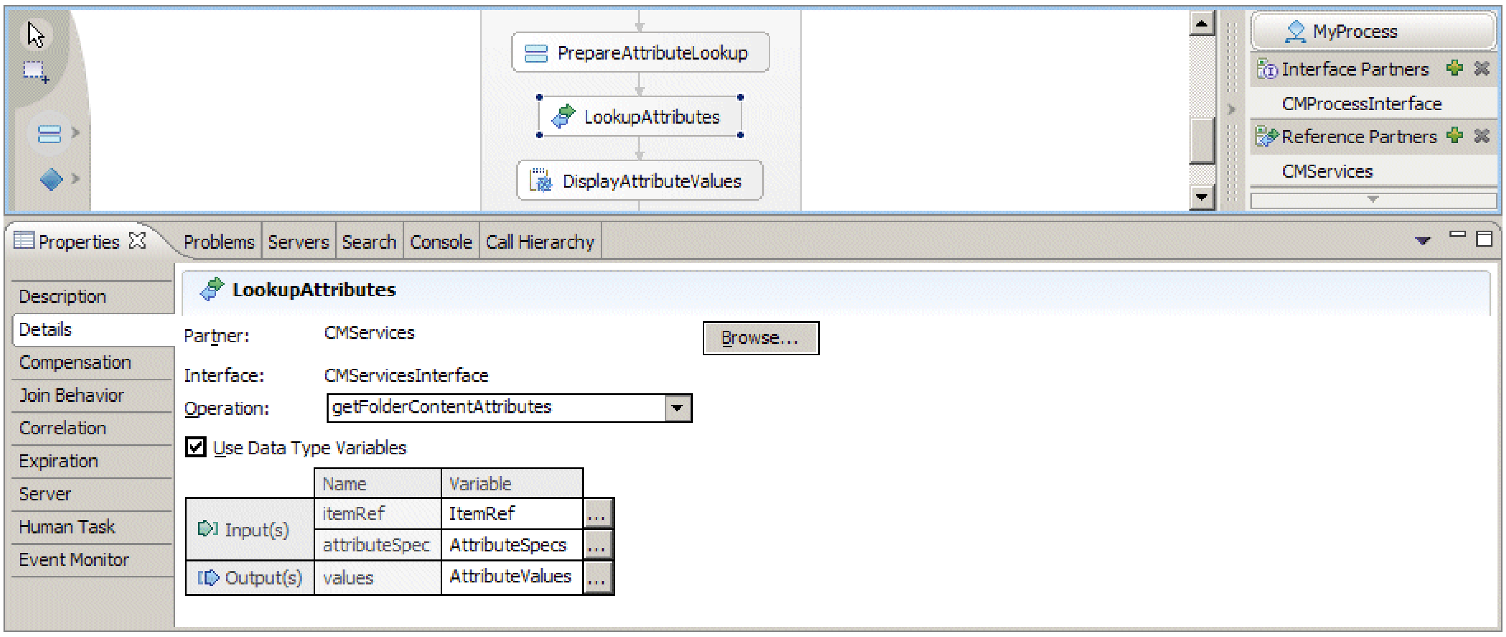
1
2

XYZ_AdjReport/XYZ_LicPlate

ItemPid : string
AttributeSpecs : AttributeSpecs
ItemRef : ItemRef
VersionNo : int
ItemId : string
ComponentId : string
AttributeValues : AttributeValues

Query: /specs[1]

The `PrepareAttributeLookup` activity assigns the two attribute specification `XYZ_AdjReport/XYZ_LicPlate` and `XYZ_AdjReport/XYZ_ClaimAmount` (not shown) to the variable `AttributeSpecs` that serves as input parameter to the 'multiple spec, multiple values' operation `getFolderContentAttributes` that looks up attribute values on items that reside in the folder.



LookupAttributes

Partner: CMServices

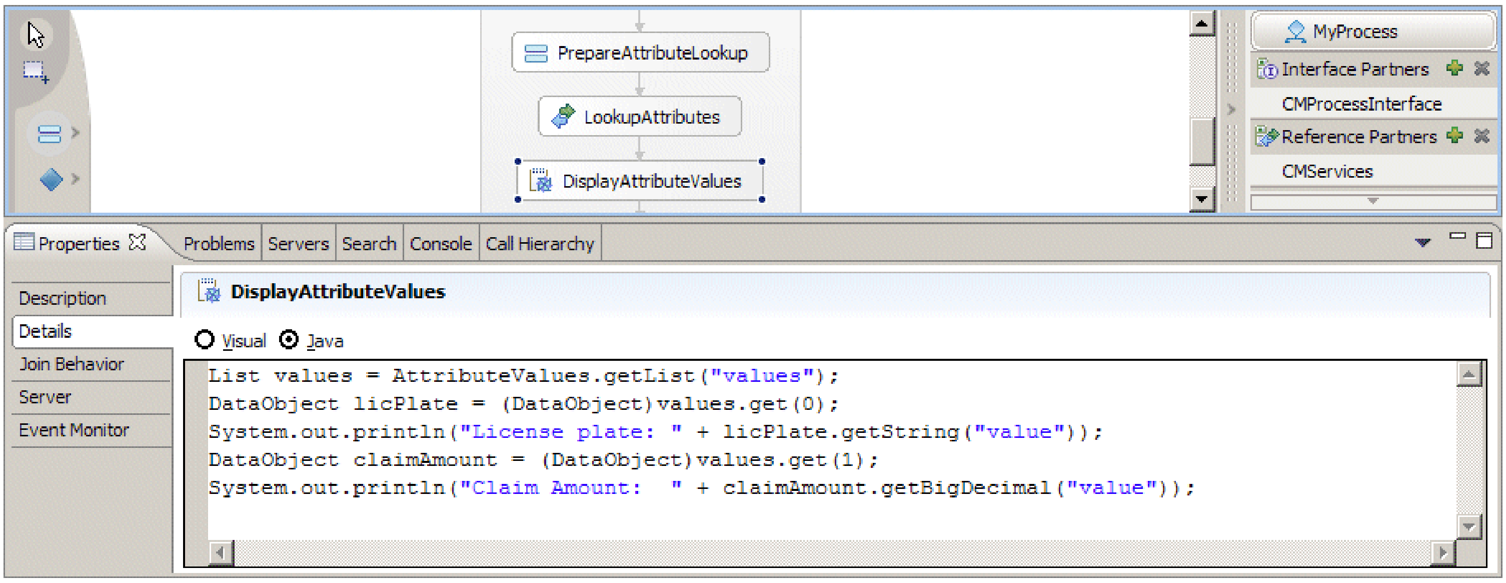
Interface: CMServicesInterface

Operation: getFolderContentAttributes

☑ Use Data Type Variables

	Name	Variable
Input(s)	itemRef	ItemRef
	attributeSpec	AttributeSpecs
Output(s)	values	AttributeValues

The result of the `getFolderContentAttributes` operation is a data object of type `AttributeValues` that contains a list of attribute values.



```
List values = AttributeValues.getList("values");
DataObject licPlate = (DataObject)values.get(0);
System.out.println("License plate: " + licPlate.getString("value"));
DataObject claimAmount = (DataObject)values.get(1);
System.out.println("Claim Amount:  " + claimAmount.getBigDecimal("value"));
```

The Java snippet `DisplayAttributeValues` retrieves the result list and prints their attribute values to the server log. Note that type conversion takes place while looking up values from the data object. It is important to understand that though an `AttributeValue` contains an element of type `anySimpleType` that corresponds to the class `Object`, it can not simply be retrieved as follows:

```
BigDecimal ca = (BigDecimal)claimAmount.get("value");
```

This leads to a class cast exception as `claimAmount.get("value")` returns a string value. Instead the following format needs to be used as in the example shown above:

```
BigDecimal ca = claimAmount.getBigDecimal("value");
```

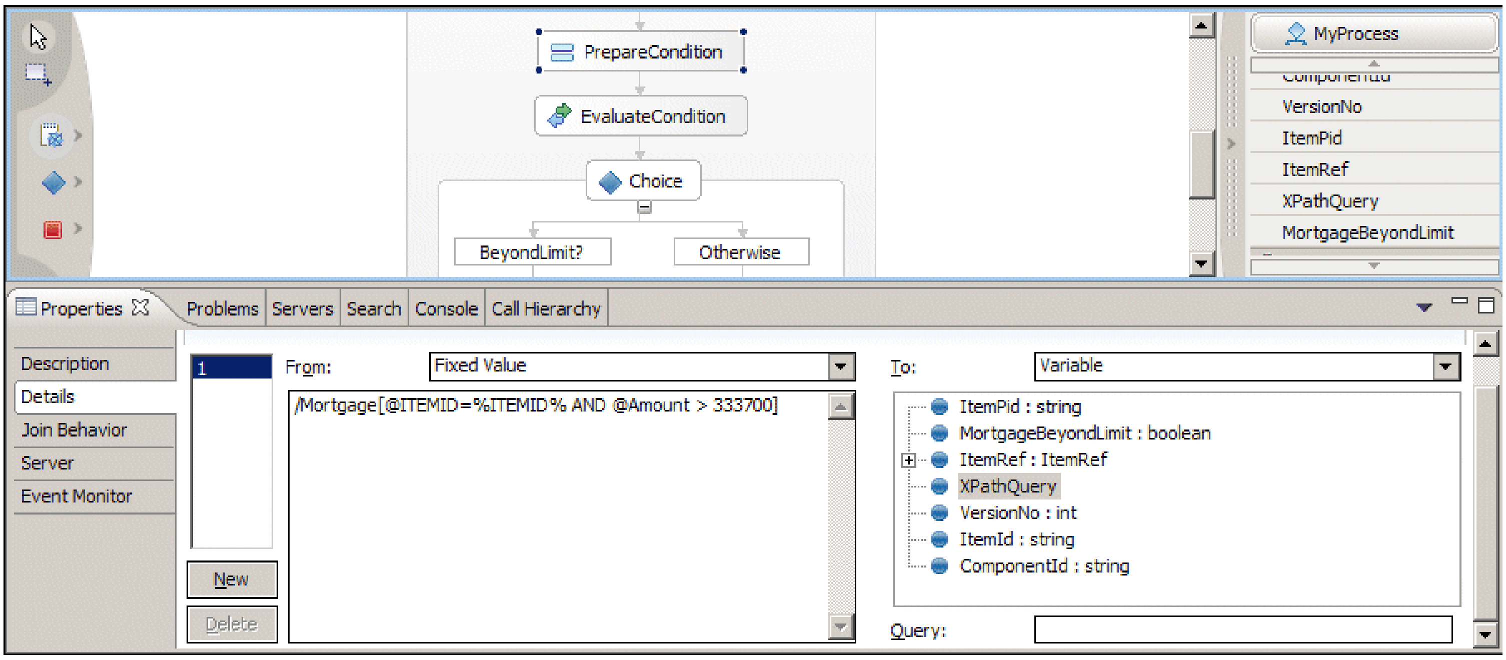
Instead of printing the values to the server log, a typical use of attribute values is in conditional processing where the path through the process is controlled by the value of attributes or variables in the process. Furthermore, the links between activities in a parallel section (flow) can be conditional, meaning that the link is only followed if the condition is true. Since the values returned from an attribute lookup are mapped to Java classes they can be used in Java expressions that form case statements or link conditions. In the example above different paths of a process can be taken by comparing the claim amount value with a predefined threshold.

Alternatively, the boolean result of the operation `evaluateCondition` can be used in conditional processing to base a decision on a content query. `evaluateCondition` takes a Content Manager XPath expression as input parameter and returns true if the result of the query is non empty. The following substitutable parameters 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 DB2 Content Manager item routed through the process. 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 conditional link expression that checks to see if the mortgage amount exceeds $333,700:



1

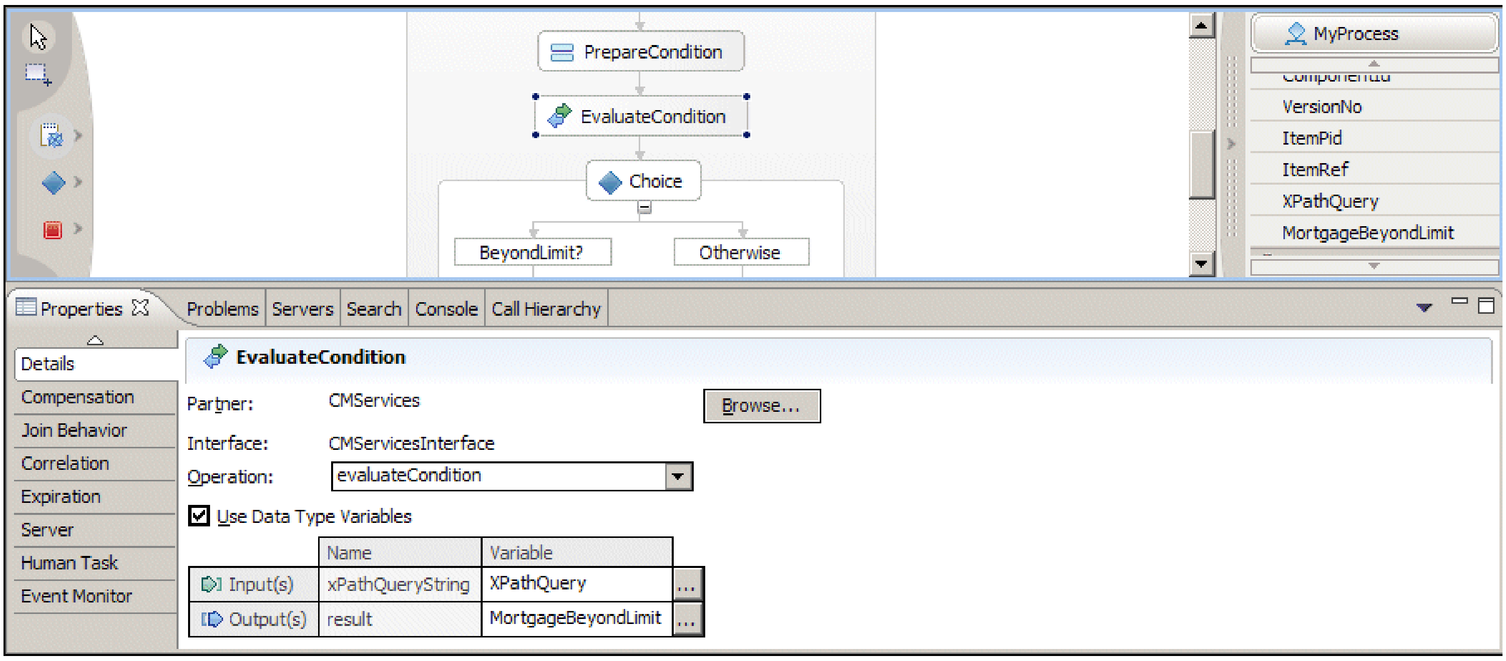
From: Fixed Value

```
/Mortgage[@ITEMID=%ITEMID% AND @Amount > 333700]
```

To: Variable

- ItemPid : string
- MortgageBeyondLimit : boolean
- ItemRef : ItemRef
- XPathQuery
- VersionNo : int
- ItemId : string
- ComponentId : string

Query:



EvaluateCondition

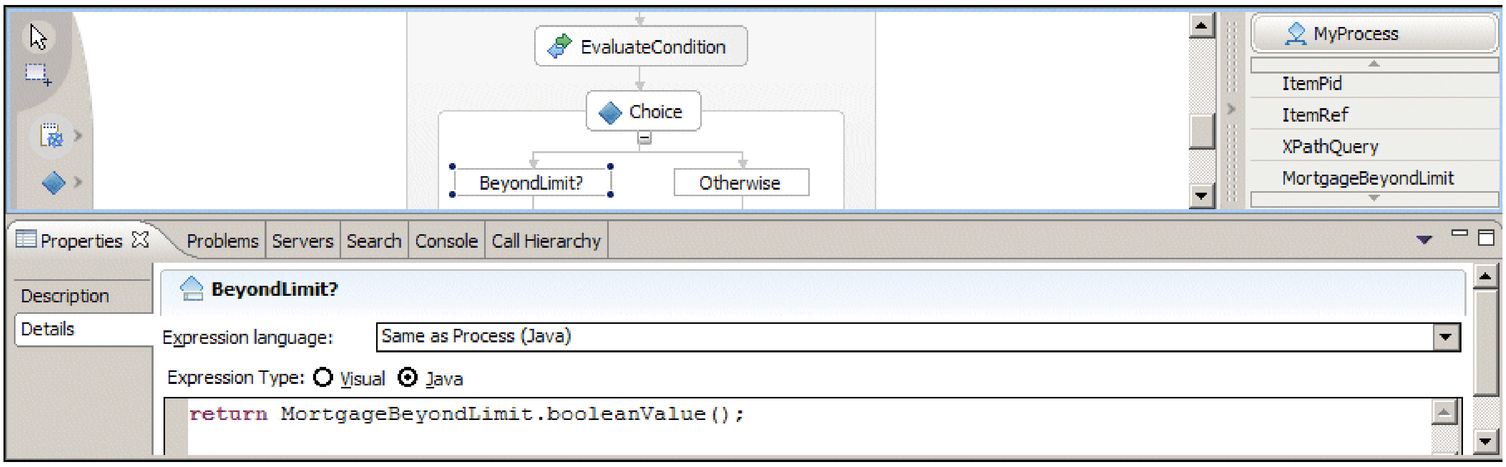
Partner: CMServices

Interface: CMServicesInterface

Operation: evaluateCondition

☑ Use Data Type Variables

	Name	Variable
Input(s)	xPathQueryString	XPathQuery
Output(s)	result	MortgageBeyondLimit



BeyondLimit?

Expression language: Same as Process (Java)

Expression Type: Visual [✓ Java]

```
return MortgageBeyondLimit.booleanValue();
```

This Java expression uses the `%ITEMID%` substitutable parameter to get the Content Manager Item ID associated with the process. This Item ID qualifies the search so that it only applies to the Content Manager item associated with the current process. In this example, the item type is Mortgage and the name of the Content Manager 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.

4.7 Additional content services

The Integration Toolkit provides some additional content services that operate on the folder which is routed through the process

Folder Services

The folder service adds one or more items to a folder. The two available operations are `addFlowItemToFolder` and `addItemToFlowFolder`. Both operations take the itemPid 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.

setACL

The `setACL` service sets the ACL of an item to a new value. If this item is a folder, it recursively changes the ACLs of all items in the folder to match this new ACL. Note that only a single level of folder-containment is supported.

The input parameters of the setACL service are the `ItemRef` of the item to be re-indexed (usually the item routed through the folder) and the name of the new ACL.

reindex

The `reindex` service assigns a new item type to an item. Note that if this service is invoked on the item that is routed through the folder, the activity that invokes it needs to be followed by `createItemRef` and `Initialize` activities as done at the beginning of a content-centric process to ensure the information about this item is consistent. For details on `createItemRef` and `Initialize` see section 4.3.

The input parameters of the reindex service are the `ItemRef` of the item to be re-indexed and the name of the target item type. The source and target item type need to be compatible for the reindex operation to suceed as specified in the DB2 Content Manager documentation. This operation has no output parameter.

4.8 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 DB2 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 available on the session bean of the Integration Toolkit which has the following interface:

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

The combinedQuery method returns a `List` of result rows. Each result row is a `List` of columns that contains the values in the order in which they are specified by the select clauses of the query (first the process properties, then the item attributes). For details on how this can be used to implement the concept of a worklist see `QuickStartClient\WebContent\listWorkItems.jsp`.

4.9 The DB2 Content Manager-based user registry

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 (task), the process management system must decide which users should have access to the item. This is referred to as staff resolution. WebSphere Process Server 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 responsible 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 Server 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. WebSphere Process Server utilizes a pluggable interface to the underlying staff repository, and comes preconfigured with interfaces for LDAP, the WebSphere User Repository and a System directory.

One of the building blocks offered by the Integration Toolkit is the *Content Manager-based user registry* which uses the DB2 Content Manager user and group definitions to support group membership. This means that users only need to be defined once, using the DB2 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.

Note: Using a custom WebSphere User Registry affects all application running on the WebSphere Application Server such as the Administrative Console. With a dedicated WebSphere Application Server installation for DB2 Content Manager and Business 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 considered.

`quickStartSampleRev2wps.zip` contains a JAR file `icmuserregistry.jar` with the CM-based user registry that is ready to be used. To install it on a WebSphere Application Server proceed as follows:

- Make sure the WebSphere Application Server on which the CM-based user registry should be installed is stopped.
- Extract `icmuserregistry.jar` from `quickStartSampleRev2wps.zip` and copy it to `<WPS_HOME>\lib`.
- Start the server.
- Start the WebSphere administrative console and log on as `icmadmin`.
- 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.
- In the **User Registries** section to the right select **Custom**.
- Enter `icmadmin` into the **Server user ID** field and the corresponding password into the **Server user password** field.
- Enter `com.ibm.bpe.cm.conmgmt.ContentManagerWebSphereUserRegistry` into the **Custom registry class name** field.
- Check **Ignore case for authorization**.
- Click **Apply**. This verifies the proper operation of the user registry. In case of a configuration problem this step fails with an error message on top of the window. If it suceeds, `icmadmin` will be able to logon the next time the server is restarted.
- To enable the runtime use of the library, a valid `bpecm.properties` file needs to be in the Application Server's properties directory. Furthermore, the `cmbsdk81.jar` library, the JDBC driver and the `<IBMCMROOT>\cmgmt` folder need to be in the class path of the application server. If you have performed the steps in Chapter 2 on this application server, this is all set up properly. If not, you need to do the following:
    - Copy the file `bpecm.properties` to the properties directory of your application server and make sure it contains the correct password of the library server administrator. See 2.4 for details.
    - Set up the runtime class path as described in 2.6.

- Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to store the updated configuration.
- Stop and restart the server to make the changes effective.

Important: When using the integrated test environment with the BPE and HTM administrator roles mapped to `AdminsGroup` a corresponding group definition needs to be created in DB2 Content Manager using the Content Manager system administration client.

If the server log periodically shows the following sequence of messages:

```
>> CM WAS User Registry - getUniqueUserId [...]
>> CM WAS User Registry - getUniqueGroupIds, user=ICMADMIN
>> CM WAS User Registry - check password for user icmadmin
Cleaning up CM connection...>> successful.
```

perform the steps described in section 6.11 to re-enable credential caching as described.

Section 6.8 describes how the user registry library can be exported from a Quick Start workspace. This may be important, e.g. if you want to replace the simple algorithm used to encrypt the password in `bpecm.properties` with a more sophisticated one. Note that the 'ROT13 encryption' only prevents the password from being obvious. This may be OK if proper file access rights are in place for the Application Server's install folders. For a real production use this method may need to be replaced with a more sophisticated encryption.

4.10 Customizing the Quick Start Client

The Quick Start Client is a generic Web-based client that does not cover the full range of capabilities offered by DB2 Content Manager and WebSphere Process Server but it provides what can be thought of as a reasonable set of typically used functions and illustrates the use of the Integration Toolkit. Since it is available as source code it can be extended and adjusted to fit the specific needs of a certain solution.

All process-specific information can be found in `QuickStartClient\JavaSource\com\ibm\bpe\cm\util\ProcessData.java`. This class needs to be modified when using the client with different roles and data. The following settings are typically subject to modification:

Global constants

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

Worklist definitions

A worklist defines certain aspects of the client's look and feel. `listWorkItems.jsp` lists the available work items based on a work list ID and `showWorkItems.jsp` displays the available actions based on this work list ID. When a user logs on to the system with a specific role, a work list ID must be provided as parameter to `listWorkItems.jsp` which it passes along to `showWorkItems.jsp`. The example customization of the Quick Start client has an initial page `index.jsp` from which users can log on to different roles of the sample process. When clicking on the map, a corresponding work list ID (WLID) is assigned so that each user sees the appropriate list of work items.

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

```
static public final int ROLE1_WLID   = 0;
static public final int ROLE2_WLID   = 1;
static public final int NUM_OF_WLIDS = 2;
static public final int DEFAULT_WLID = ROLE1_WLID;
```

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

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

```
addToFolderActionItemTypes[ROLE1_WLID] = { "AdjReport", "FraudReport",
"ReliabilityStudy" };
addToFolderActionNames[ROLE1_WLID]     = { "Adjuster Report", "Fraud Report",
"Reliability Study" };
```

...means that a user with WLID `ROLE1_WLID` may add instances of the item types `djReport`, `FraudReport`, and `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`.

Staff conventions

In- and output messages of participating tasks are implemented based on input and output data objects assigned to the staff operation in a WSDL file. We recommend a naming convention such as: `<TaskName>TaskInterface.wsdl` for the WSDL and `<TaskName>Request.xsd` for the input data object and `<TaskName>Response.xsd` for the output data object. The name of the input parameter in the WSDL should be `request`, the name of the output parameter `response`. `JavaSource\com\ibm\bpe\cm\CompleteWorkServlet.java` relies on this naming convention. If all properties of the output data object are of type String, this servlet automatically maps parameters of the HTML form to the corresponding properties of the output data object if the parameter name is identical with the name of the property. The WSDL and XSD files are stored in `QuickStartClient\WebContent\WEB-INF\lib` so they are in the runtime classpath of the Web application.

The method `staffActions(String activityName, DataObject inDO)` 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 properties of the output data object (`<TaskName>Response.xsd`) are of type String, no custom logic needs to be added to `JavaSource\com\ibm\bpe\cm\CompleteWorkServlet.java` as it automatically maps form values to output properties with the same name. If non-string-typed variables are involved, an additional conversion step is required in the process or custom logic needs to be added to this servlet.

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

```
static public String staffActions(String activityName, DataObject inDO) {

   try {
     String result = "<td colspan=\"4\"></td>";   // default

     if (activityName.equals("Review")) {
            String textToBeReviewed = inDO.getString("TextToBeReviewed");
            result = "<TD colspan=\"2\">\n" +
                              "Text to be reviewed:" + textToBeReviewed +
                 "</TD>" +
                                 "<TD colspan=\"2\">\n" +
                 "<SELECT name=\"msgpart.rating\">" +
                                 "<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;
}
```

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 .

Limitations of the QuickStartClient

Since the QuickStartClient is a sample, it is not considered a complete Content Manager or Process Choreographer client. However, as the this section should have demonstrated, it is not specific to the sample process but can be easily modified to support other content and process models. Individual functions can be added by extending the available source code. Here is a list of some of the client's limitations.

- Checkin / -out of Content Manager items can not be done explicitly (it is done implicitly when e.g. modifying them).
- There is no general Content Manager or Process Choreographer search capability. The only search capability that is available is implicit in the worklist creation (combined query).
- Items in folders are listed as they are retrieved from Content Manager. No special sort order is applied. However, this can easily be added based on the Content Manager method `sortFolderItmes()` of the `CMBItem` class.
- Child items may be created but the client currently does not provide a 'delete child' option.
- When adding multiple parts to an item, at most one part may be edited in the sense of adding annotations. This is not a problem when using the Document class (as supported by the Content Manager clients) since it restricts the use of parts to one per item.

4.11 Considerations for production use

Chapter 2 explained the steps required to set up a test environment. This section briefly covers some aspects that are important when planning to move to a production environment.

Globalization and Accessibility

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` or `ConnectionException`.

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

Scalability and Performance

For long-running processes in a production environment we recommend to consider replacing the use of Cloudscape as platform for the BPE database with an enterprise-level database such as DB2 UDB. See the WebSphere Process Server documentation for details on how this is done.

To avoid the server termination problem apply the steps in 6.4. Disabling the just in time compiler is not an option in a production environment.

Security and User Management

For evaluation purposes the development environment set up in chapter 2 used a file based custom registry. However, in a production environment this is not recommended as it relies on a file that contains passwords in readable format. For use in a production environment we recommend to use either the Content Manager-based user registry or an LDAP directory server. The steps required to use the Content Manager-based user registry are described in section 4.9 and 6.8. Details on using an LDAP server can be found in the product documentation referenced at the end of this document.

It may be advisable to consider implementing a 'real' encryption method for the password stored in `bpecm.properties`. The sample offers a simple encryption method referred to as ROT13 which prevents the password from being obvious but this does not offer a real protection in cases where access to the Process Server's lib directory is not (or can not be) properly protected. `CMConnectionManagement\com\ibm\bpe\cm\conmgmt\Environment.java` supports the extension with stronger encryption mechanisms based on the configuration property `ContentManagerAuthenticationPwdEncryption`.

4.12 The project structure of a custom integration solution

To create a custom integration solution the following projects need to be imported from `quickStartSampleRev2wps.zip`:

- CMBPCIntegrationLibrary
- CMBPCIntegration
- CMBPCIntegrationBeans
- CMConnectionManagement
- QuickStartClient
- QuickStartClientApp

The folder `QuickStartClient\WebContent\WEB-INF\lib` that is in the runtime classpath of the Quick Start Client needs to include the following resources: copies of `bpe137650.jar`, `task137650.jar` from `<WPS_ROOT>\ProcessChoreographer\client`, `bpecmutil.jar` from the same directory of the Integration Quick Start, and `CMProcessInterface.wsdl` from `CMBPCIntegrationLibrary\com\ibm\bpe\cm\`. Furthermore the WSDL and XSD files for the task interfaces are to be placed into this directory.

Create a process as described in section 4.3 adding functionality of the Integration Toolkit as described in the respective sections of this chapter. As tasks are added, adjust the definitions in `QuickStartClient\JavaSource\com\ibm\bpe\cm\utils\ProcessData.java` accordingly. For testing purposes it may be useful to create a modified `QuickStartClient\WebContent\index.jsp` that allows logging on to the process with different roles or starting the process either explicitly (using a query) or implicitly (using Content Event Handling).

It may be useful to disable the feature **Automatically delete the process after completion** during testing as this simplifies investigating error conditions since failed work items remain the process database and can be retrieved using the BPC Explorer. Note that if the process contains a collection point or one or more human tasks it needs to be defined as **long running**. Both settings, **Process is long-running** and **Automatically delete...** can be made on the **Details** tab of the process properties view. To switch to this view click the white background of the process definition in the BPEL editor and click the **Properties** tab. Click **Details** to switch to the details view. This displays two check boxes with which these properties can be dis- or enabled.

5. Invoking DB2 Content Manager Web Services from within a process

Service invocation is one of the key integration patterns in service orientation architectures (SOA). DB2 Content Manager V8.3 offers a powerful web service interface and thereby adds a standards-based access to the content repository which can be invoked from any service consumer. WebSphere Process Server V6 comes with a BPEL compliant business process engine (Business Process Choreographer) that allows orchestrating and choreographing (web) service calls. This chapter explains the steps needed to invoke DB2 Content Manager web services from a BPEL process and how to build an integrated solution based on web service technology.

Here is an outline of the scenario used in this chapter:

- Prompt a user for a DB2 Content Manager XPath query expression based on the `XYZ_AdjReport` item type of the sample data.
- Pass the expression as an input message into a BPEL business process.
- In the business process, invoke the Content Manager **RunQuery** web service.
- Processes the result set in the BPEL business process.
- Print the values of the first name and last name attributes for each adjuster report in the result set.

Important: If you want to use both the Content Manager Web Services and the Integration Toolkit you need to perform the steps in 6.10 to avoid a conflict with libraries in the classpath.

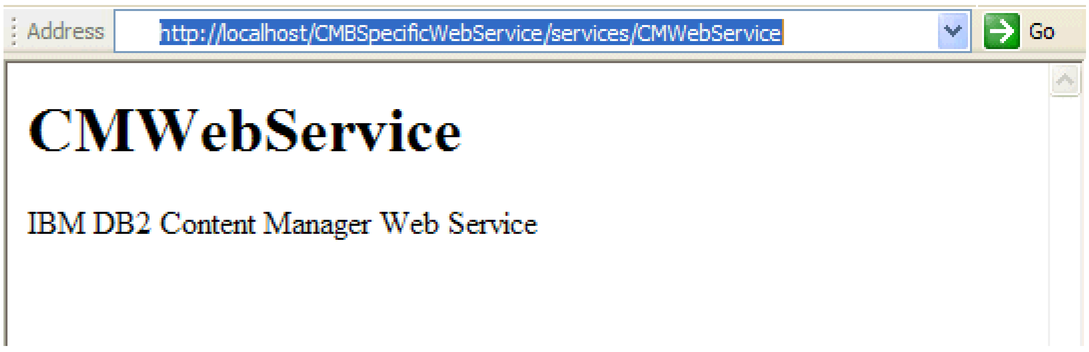
5.1 Setting up and configuring the system environment

This chapter assumes the following system configuration:

a. Development workstation A hosts:
   - *WebSphere Integration Developer V 6.0.1 optionally with the WebSphere Test Environment (WTE)*
   - *If the WTE is not installed: WebSphere Process Server V6.0.1*
   - *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*
   - *DB2 Information Integrator for Content V8.3* - Web services support

Details on how to install the DB2 Content Manager web service support can be found in the product documentation (see Further Reading). The Content Manager web services support installs an additional server `cmwebsvc` on the WebSphere Application Server. To perform the scenario described in this chapter the DB2 Content Manager web service server must be properly installed and running. To verify this point your browser to the URL shown below. Note that you may need to replace `localhost` with the name of the repository server and and you may need to add a port number.


Address
http://localhost/CMBSpecificWebService/services/CMWebService
Go

CMWebService

IBM DB2 Content Manager Web Service

To use the DB2 Content Manager web services in a Process Server environment, the following step needs to be performed to ensure proper treatment of namespaces.

- Open the file `<IBMCMROOT>\cmgmt\cmbxmlservices.properties` in an editor and add the line

```
forceItemNamespace=http://www.ibm.com/xmlns/db2/cm/beans/1.0/schema
```

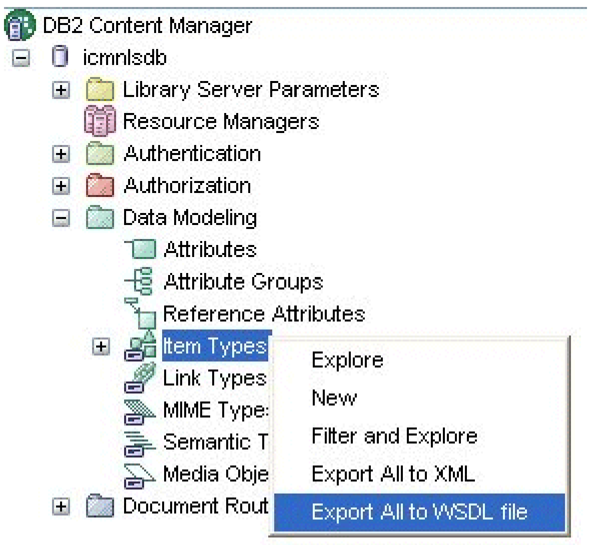
Make sure you have installed WebSphere Integration Developer 6.0.1 with the WebSphere Test Environment (WTE) or an additional standalone WebSphere Process Server 6.0.1. No additional setup or configuration steps are required to work with the DB2 Content Manager web services.

5.2 Exporting the web service and data definition

In order to invoke a DB2 Content Manager web service, you need to export a WSDL package from the System Administration Client that contains WSDL and XSD files with information about...

- Where the web service server is installed
- Which operations are available
- Which data types are available

Item type and attribute definitions are important ingredients of a DB2 Content Manager data model. When exporting the data model or parts of it to a WSDL package, the data definitions are translated to XSD definitions and thereby made available at the web service interface. If you plan to access your item type definition in a business process, make sure you defined them in DB2 Content Manager prior to exporting the web service definition. To export a complete data model into a WSDL package log on to the System Administration Client, select **Data Modeling - Item Types** and then right-click to bring up the pop-up menu. Select **Export All to WSDL file**. You can also export item types individually or subsets of item types by clicking them and selecting **Export to WSDL file**.



Specify a location for the resulting zip file. Then, extract the zip file to a location in your file system. As a result, you get the following four files:

- `CMWebService.wsdl`
- `cmbmessages_modified.xsd`
- `cmdatamodel_modified.xsd`
- `itemtype_modified.xsd`

Important: ensure that the port number and host namematches your settings of the `cmwebsvc` service. This information can be found near to the end of `CMWebService.wsdl`:

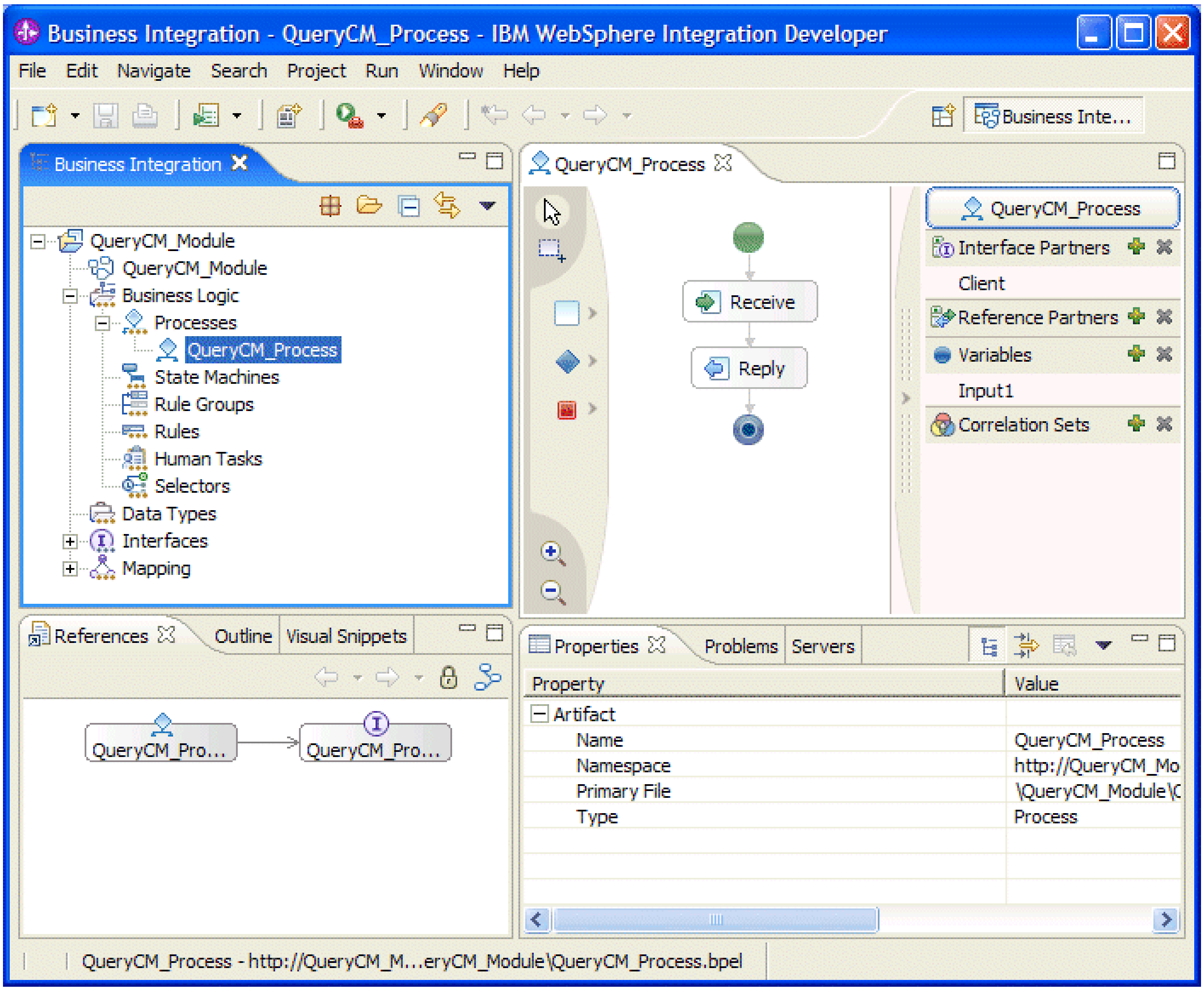
```
        <wsdl:port binding="tns:CMWebServiceBinding" name="CMWebServicePort">
            <soap:address
                location="http://localhost:9081/CMBSpecificWebService/services/
CMWebService"/>
        </wsdl:port>
```

5.3 Creating a new Project in WebSphere Integration Developer

Start WebSphere Integration Developer 6.0.1 (WID), point the workbench to a new empty workspace, and proceed as follows:

1. Right-click in the **Business Integration** view and select **New > Module**. This opens the **New Module** window.
2. Enter the module name `QueryCM_Module` and click **Finish.**
3. Expand **Business Logic**, right-click **Processes** and select **New > Business Process**. The **New Business Process** window opens.
4. Enter `QueryCM_Process` into the **name** field and click **Finish**. This creates the artifacts for a new business process based on a simple interface with a single input variable `Input1` of type **String**. This variable will be used later to specify a DB2 Content Manager XPath query string.

When the process generation step is completed the WID screen look as follows:



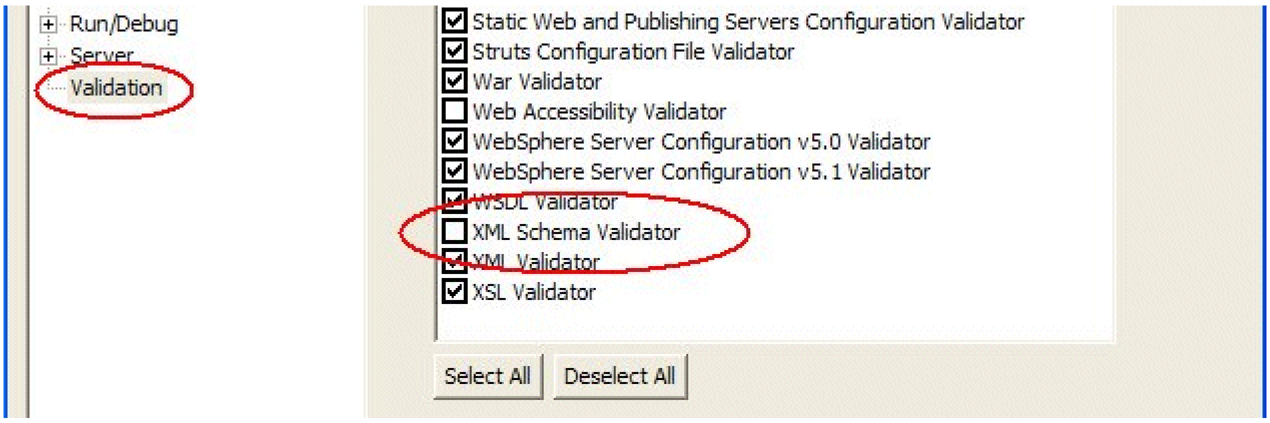
Property	Value
Artifact	
Name	QueryCM_Process
Namespace	http://QueryCM_Mo
Primary File	\QueryCM_Module\C
Type	Process

QueryCM_Process - http://QueryCM_M...eryCM_Module\QueryCM_Process.bpel

5.4 Importing Web Service and Data Definition

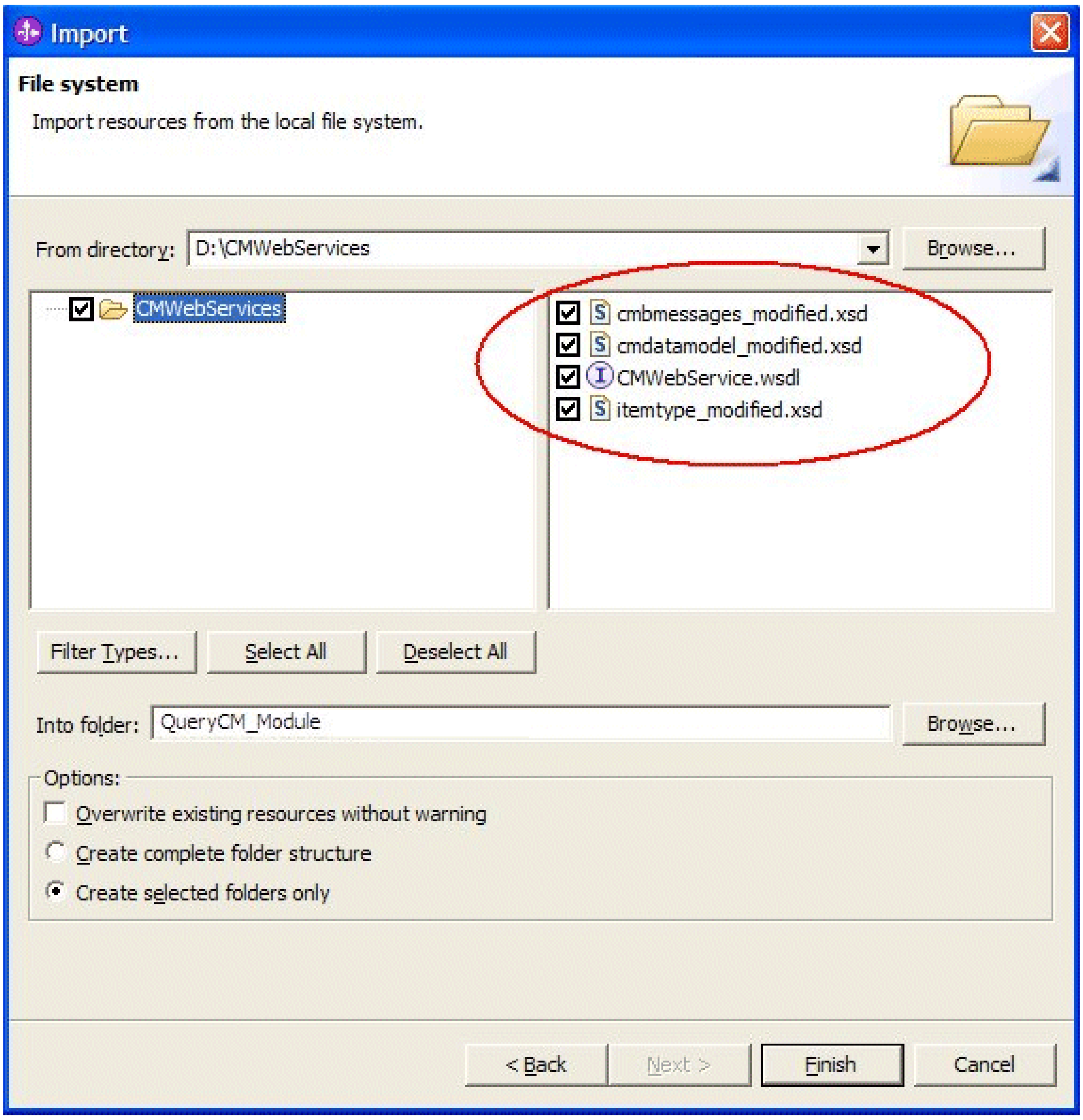
In order to work with the DB2 Content Manager web services, you need to import both the web service definition and the data definitions (based on the item types) to WebSphere Integration Developer.

Note: You might have to disable XML schema validation before importing the Content Manager WSDL and XSD files. Select **Window > Preferences > Validation** and disable the **XML Schema Validator** checkbox.



- [x] Static Web and Publishing Servers Configuration Validator
- [x] Struts Configuration File Validator
- [x] War Validator
- [ ] Web Accessibility Validator
- [x] WebSphere Server Configuration v5.0 Validator
- [x] WebSphere Server Configuration v5.1 Validator
- [x] WSDL Validator
- [ ] XML Schema Validator
- [x] XML Validator
- [x] XSL Validator

1. Right-click in the **Business Integration** view and select the **Import …** command from the pop-up menu.
2. On the **Import** window select **File system** and specify the file system location where you previously stored the WSDL and XSD files that you have exported from the DB2 Content Manager System Administration Client.
3. Click **Select all** and **Finish**.



Import

File system

Import resources from the local file system.

From directory: D:\CMWebServices

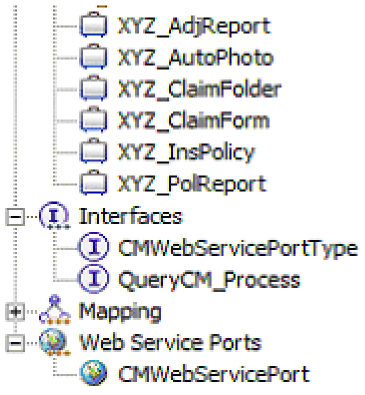
☑ CMWebServices

☑ cmbmessages_modified.xsd
☑ cmdatamodel_modified.xsd
☑ CMWebService.wsdl
☑ itemtype_modified.xsd

Into folder: QueryCM_Module

Options:
☐ Overwrite existing resources without warning
○ Create complete folder structure
● Create selected folders only

Explore the imported definitions in the **Data Object**, **Interface** and **Web Service Ports** sections of the **Business Integration** view.



- XYZ_AdjReport
- XYZ_AutoPhoto
- XYZ_ClaimFolder
- XYZ_ClaimForm
- XYZ_InsPolicy
- XYZ_PolReport
- Interfaces
  - CMWebServicePortType
  - QueryCM_Process
- Mapping
- Web Service Ports
  - CMWebServicePort

5.5 Modeling the BPEL process to invoke the DB2 Content Manager Web Services

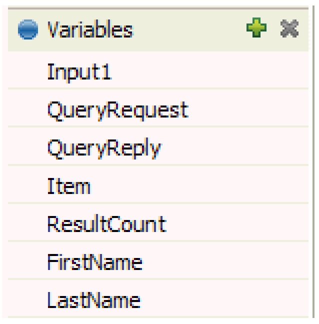
In order to run our scenario, we need input and output messages for the **RunQuery** web service call. For result processing in a loop, we also need a counter and a variable to store an item from the result set.

1. Define process variables based on the table below

Variable name	Data Type
`QueryRequest`	`RunQueryRequest (CM definition)`

`QueryReply`	`RunQueryReply (CM definition)`
`Item`	`Item (CM definition)`
`ResultCount`	`int`
`FirstName`	`string`
`LastName`	`string`

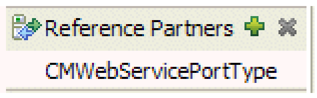
- Click the **+** in the **Variables** section of the BPEL editor to create a new variable named `Variable`.
- Replace the default name `Variable` with the corresponding name from the table below.


Variables
Input1
QueryRequest
QueryReply
Item
ResultCount
FirstName
LastName

- While the variable name is selected, click the **Details** tab in the **Properties** view (Note that you need to perform this only once as this tab stays selected in the view while adding variables).
- Click **Browse** and select the corresponding type as specified in the table above.

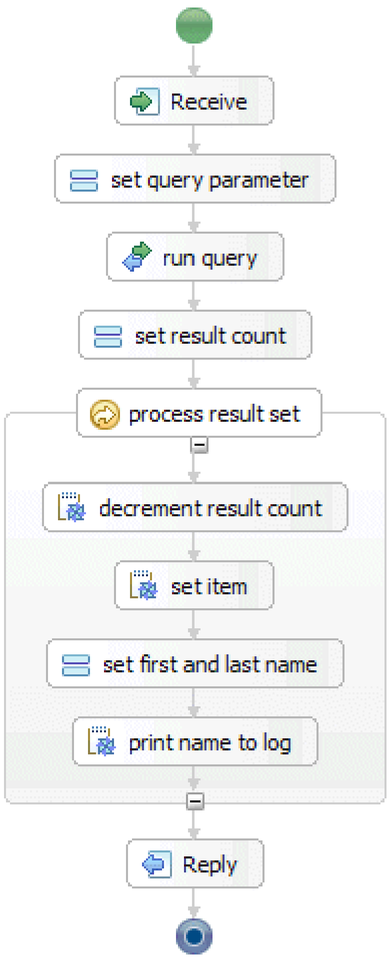
2. Define a new partner for the BPEL process

- In the **Business Integration** view expand **Interfaces**.
- Drag the `CMWebServicePortType` interface definition from the **Business Integration** view and drop it to the **Reference Partners** section of the BPEL editor. This creates a new partner which can be referenced by invoke activities of the BPEL process.


Reference Partners
CMWebServicePortType

3. Define the BPEL Process

Model the BPEL process according to the following layout by selecting activity types from the menu on the left of the BPEL editor and dropping them onto the appropriate location of the business process. While the activity stays selected, click the **Description** tab of the **Properties** view to enter the activity name.



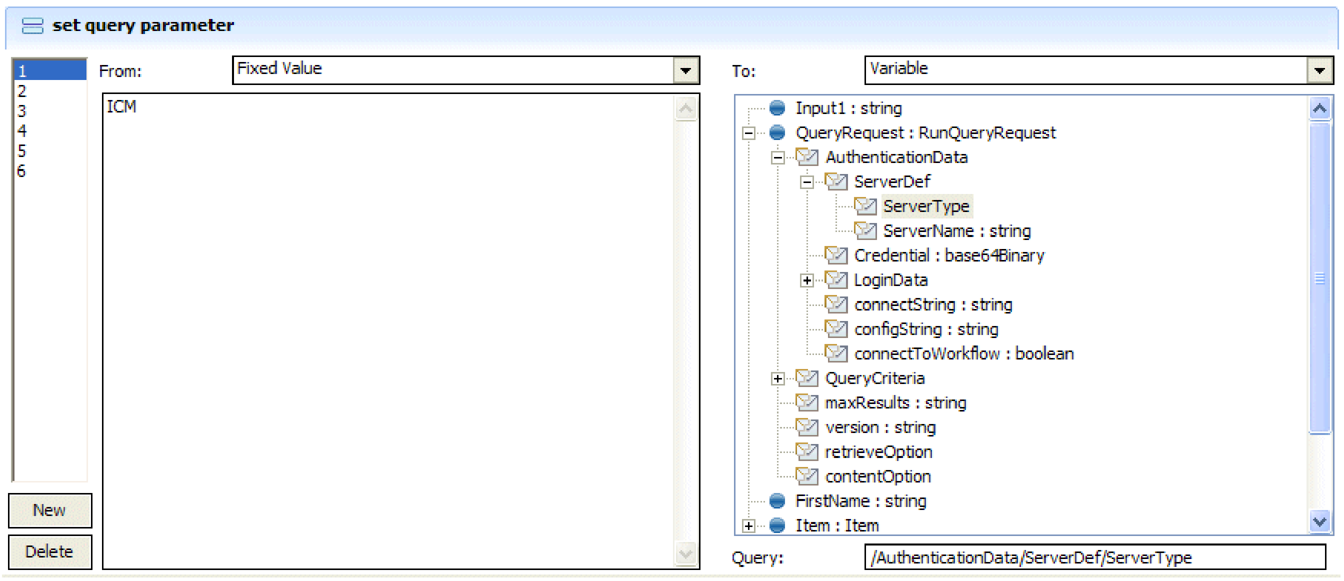
Note that this is an example of a non-interruptible process (micro flow) since it does not involve human activities (tasks) and potentially long-running services such as a collection point. Non-interruptible processes do not need to store information about intermediate process states in the process engine's database and run in a single transaction. If a process does not qualify as non-interruptable, this needs to be explicitly set by right-clicking the background of the BPEL editor and selecting **Process is long-running** in the **Details** pane.

4. Further specify the activities as follows:

   - Initiating **Receive** activity
     receive is an initiating receive activity which takes the value of variable `Input1` (as a String). No additional parameters need to be set - just use the **Receive** activity which the tools create for you.

   - **set query parameter** activity
     set query parameter is an assign activity which sets all attributes required to invoke the web service. Use the **Details** panel in the **Properties** tab to set the following values for variable **QueryRequest** as shown below:

"From" parameter type	"From" parameter value	"To" parameter type	"To" parameter - **QueryRequest**
Fixed Value	`ICM`	Variable	`/AuthenticationData/ServerDef/ServerType`
Fixed Value	`icmnlsdb`	Variable	`/AuthenticationData/ServerDef/ServerName`
Fixed Value	`icmadmin`	Variable	`/AuthenticationData/LoginData/UserID`
Fixed Value	`<password>`	Variable	`/AuthenticationData/LoginData/Password`
Variable	`Input1`	Variable	`/QueryCriteria/QueryString`
Fixed Value	`ATTRONLY`	Variable	`/retrieveOption`


set query parameter

From: Fixed Value

ICM

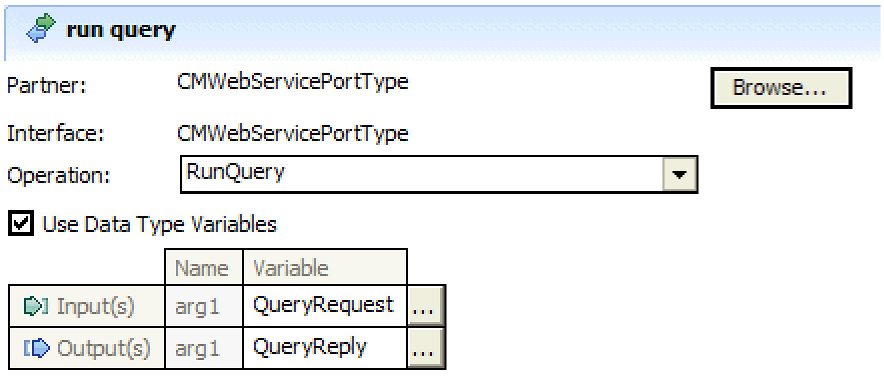
To: Variable

- Input1 : string
- QueryRequest : RunQueryRequest
  - AuthenticationData
    - ServerDef
      - ServerType
      - ServerName : string
    - Credential : base64Binary
    - LoginData
    - connectString : string
    - configString : string
    - connectToWorkflow : boolean
  - QueryCriteria
  - maxResults : string
  - version : string
  - retrieveOption
  - contentOption
- FirstName : string
- Item : Item

Query: /AuthenticationData/ServerDef/ServerType

- **run query** activity
  run query is an invoke activity which uses the `CMWebServicePortType` partner link to invoke the DB2 Content Manager **RunQuery** web service.
  Use the **Details** panel in the **Properties** tab to associate the invoke activity with the `CMWebServicePortType` partner:

  1. Click **Browse** and select the `CMWebServicePortType` from the list of partners. Click **OK**.
  2. Specify the **Operation**: select **RunQuery** from the list of available operations.
  3. Ensure **Use Data Type Variables** is selected.
  4. Click **...** in the right-most column of the **Input(s)** row and select `QueryRequest` as input variable.
  5. Click **...** in the right-most column of the **Output(s)** row and select `QueryReply` as output variable.


run query

Partner: CMWebServicePortType

Interface: CMWebServicePortType

Operation: RunQuery

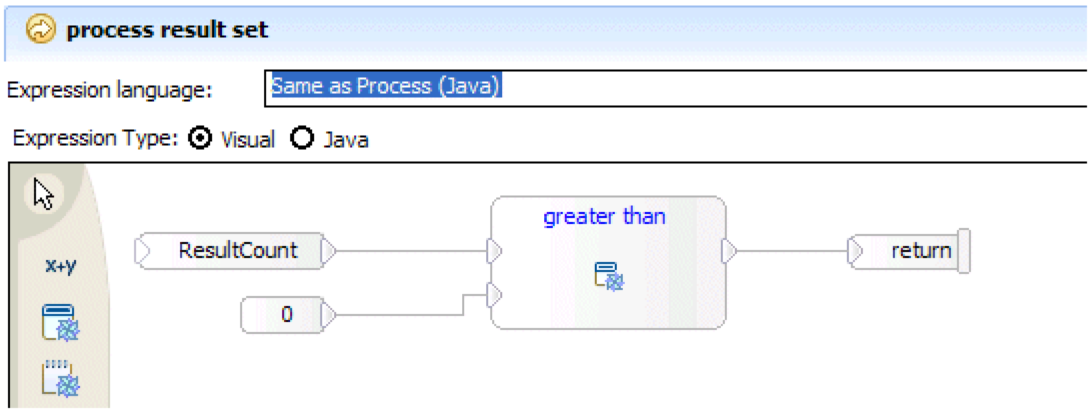
[✓] Use Data Type Variables

	Name	Variable
Input(s)	arg1	QueryRequest
Output(s)	arg1	QueryReply

- **set result count** activity
  set result count is an assign activity which extracts the result count value from the query result to a process variable `ResultCount`. Use the details panel to copy `/ResultSet/count` in `QueryReply` to variable `ResultCount`.

- **process result set** activity
  process result set is a loop activity which executes an associated block of activities multiple times as long as **ResultCount** is larger than 0. Result count is also decremented by 1 in this activity. You can use the visual editor to compare the value of the **ResultCount** variable with 0 as follows:
  1. Select the activity and click **Details** in the **Properties** view.
  2. Select **Same as process (Java)** from the list of Expression Languages. Ensure the expression type is set to **Visual**.
  3. Delete the input element named **false**.
  4. Click the symbol for **Standard Visual Snippets**, select **math > greater than** and click **OK** to insert the corresponding visual snippet into the drawing area.
  5. Drag the `ResultCount` variable from the list to the right into the drawing area.
  6. Click the symbol **x+y** to create a new expression. Right click it and enter `0`.

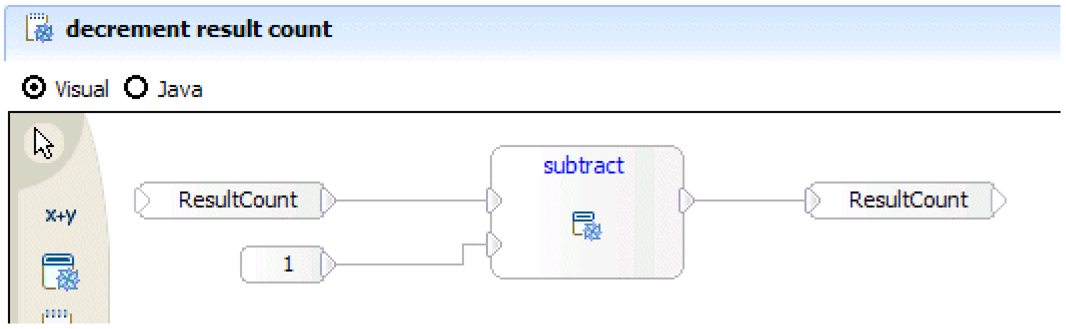
7. Wire the snippet representing the `ResultCount` variable to the first and the snippet representing the constant 0 to the second input slot of the comparison operator and wire the output slot of the comparison to snippet labeled **Result**.


process result set

Expression language: Same as Process (Java)

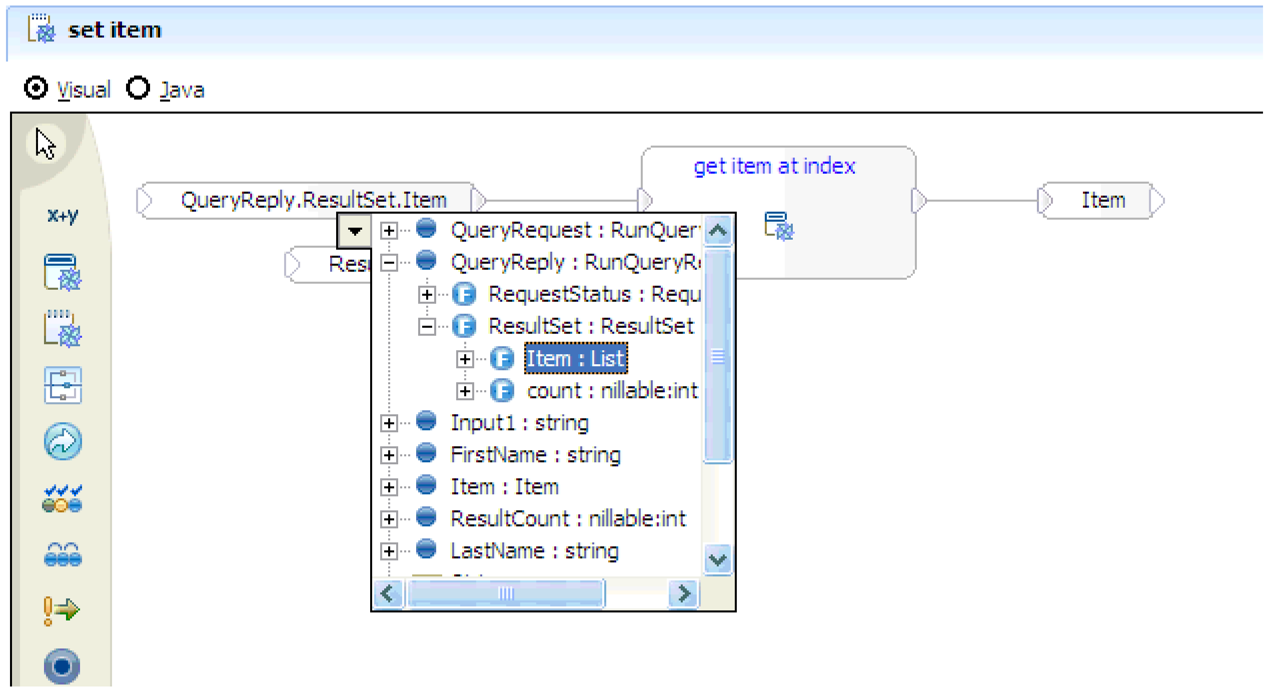
Expression Type: [✓ Visual] Java

- **decrement result count** activity
  You have to decrement the result count variable with each run of the loop.
  Proceed as in the previous step to combine the visual snippets needed to subtract **1** from the **ResultCount** variable:

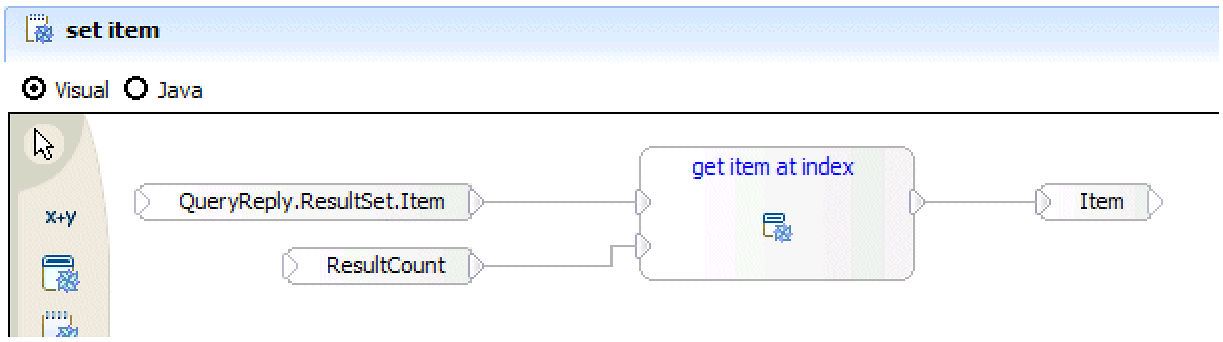

decrement result count

[✓ Visual] Java

- **set item** activity
  set item is a visual snippet activity which copies one item from the result set for further processing.
  This is the first activity inside the loop. Use a visual snippet as in the preceding steps to extract a single item from the result list (according to the index value stored in ResultCount). Note that the **get item at index** snippet is located in the **list** package of the standard visual snippet library.


set item

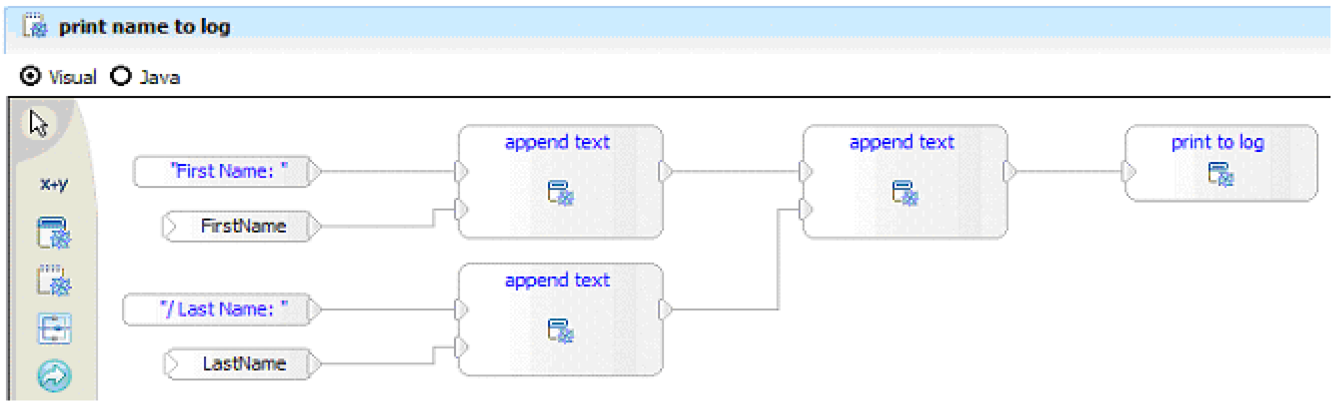
[✓ Visual] Java



- **set first and last name** activity
  set first and last name is an assign activity which extracts the adjuster's first and last name from the item to dedicated string variables (FirstName and LastName respectively).
  Use the **Details** panel in the **Properties** tab to set the following values from **Item** to for variables `FirstName` and `LastName`.

"From" variable **Item**	"To" variable
`/ItemXML/XYZ_AdjReport/XYZ_AdjustFName`	`FirstName`
`/ItemXML/XYZ_AdjReport/XYZ_ClaimLName`	`LastName`

- **print name to log** activity
  print name to log is again a visual snippet which prints (for demo purposes) the adjuster's first and last name (stored in the BPEL variables) to the Application Server's log file. Note that the **append text** snippet is located in the **text** package and **print to log** in the **utility** package of the standard visual snippet library.

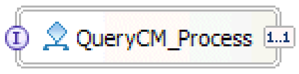


5. Press Ctrl-S in the BPEL editor to store the process definition. Note that this builds the workspace in the background.

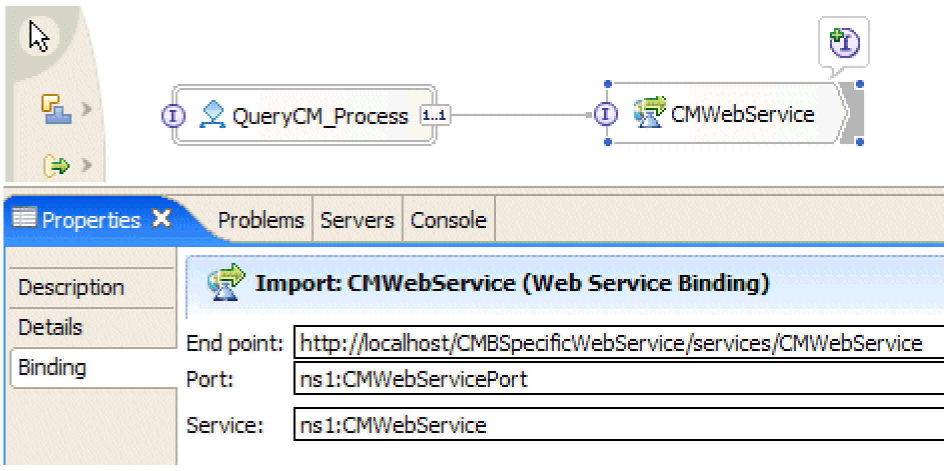
5.6 Assembling the module and testing the sample process

The `QueryCM_Module` will now be completed by connecting the sample process with the required partner. Then we describe how the business process can be tested using the BPC Explorer.

1. Double click **QueryCM_Module** right above the **Business Logic** in the **Business Integration** view to open the assembly editor.
2. Drag and drop the BPEL process `QueryCM_Process` you just modeled into the editor view.



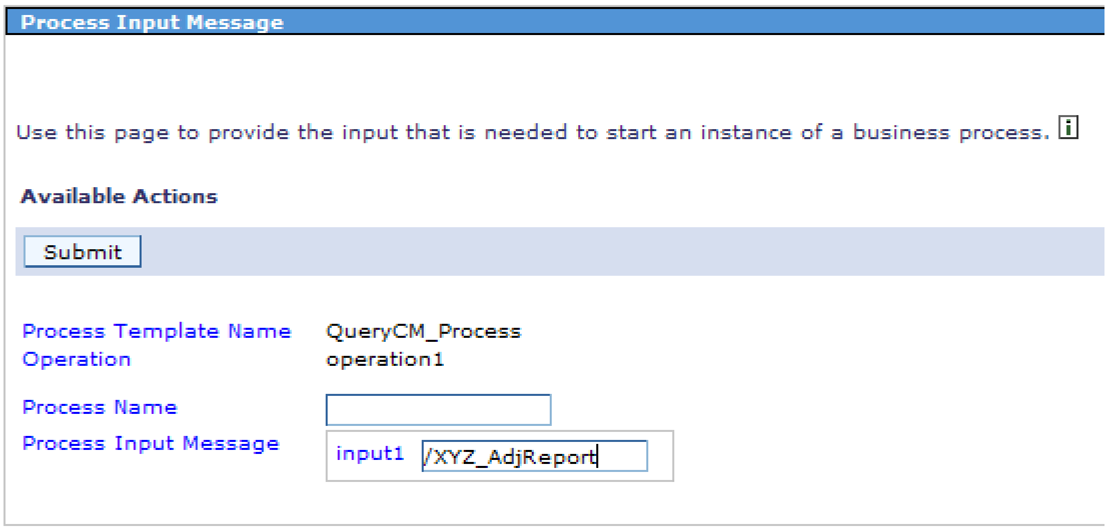
3. Drag `CMWebServicePortType` from the **Business Integration** view and drop it into the assembly area. The **Component Creation** window opens.
4. Select **Import with Web Service Binding** and click **OK**. This creates a new import element with the appropriate interface and binding.
5. With the new import element selected click the **Description** tab in the **Properties** view and enter `CMWebService` into the **name** field.
6. Right-click any of the two elements and select **Wire to existing** to wire the process with its referenced partner.



Import: CMWebService (Web Service Binding)

End point: http://localhost/CMBSpecificWebService/services/CMWebService
Port: ns1:CMWebServicePort
Service: ns1:CMWebService

Now that the module creation has been completed the new application is ready to be deployed and tested. Create a new server configuration pointing to a WebSphere Process Server installation and add your module to the server (right-click the server configuration, select **Add and remove projects**, and add `QueryCM_ModuleApp` to the list of configured projects). Start the server and use the BPC explorer to submit a new request to your business process (right-click the server configuration and select **Launch > BPC Explorer**). In the BPC Explorer start a new Process (click on **My Process Templates**, select QueryCM_ Process and click on **Start Instance**). To test your business process, add `/XYZ_AdjReport` as the input string for the input message. This will return all adjuster report data stored with the DB2 Content Manager first steps sample data.



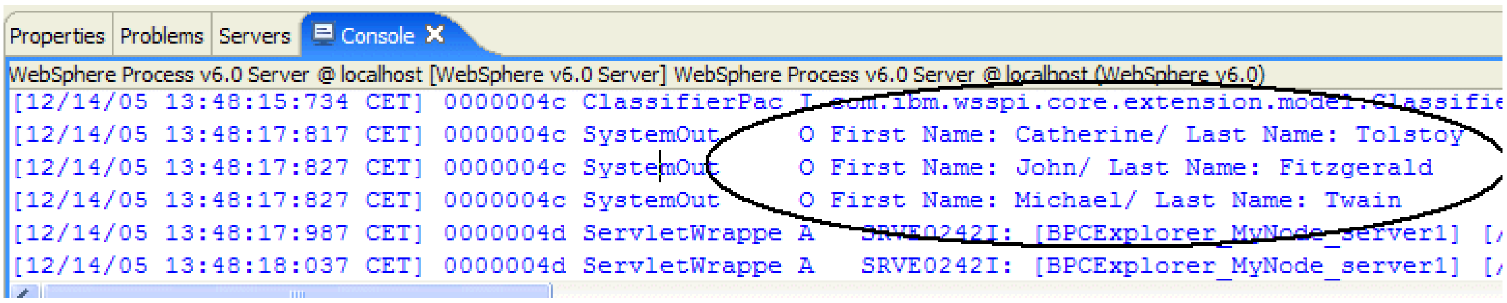
Process Input Message

Use this page to provide the input that is needed to start an instance of a business process.

Available Actions

Process Template Name QueryCM_Process
Operation operation1
Process Name
Process Input Message input1 /XYZ_AdjReport

The console output created by the BPEL process looks as follows:



```
WebSphere Process v6.0 Server @ localhost [WebSphere v6.0 Server] WebSphere Process v6.0 Server @ localhost (WebSphere v6.0)
[12/14/05 13:48:15:734 CET] 0000004c ClassifierPac I  com.ibm.wsspi.core.extension.model.Classifie
[12/14/05 13:48:17:817 CET] 0000004c SystemOut     O  First Name: Catherine/ Last Name: Tolstoy
[12/14/05 13:48:17:827 CET] 0000004c SystemOut     O  First Name: John/ Last Name: Fitzgerald
[12/14/05 13:48:17:827 CET] 0000004c SystemOut     O  First Name: Michael/ Last Name: Twain
[12/14/05 13:48:17:987 CET] 0000004d ServletWrappe A   SRVE0242I: [BPCExplorer_MyNode_server1] [/
[12/14/05 13:48:18:037 CET] 0000004d ServletWrappe A   SRVE0242I: [BPCExplorer_MyNode_server1] [/
```

Please note that for simplicity of this description, no error handling has been implemented. In particular, the business process logic assumes that the Content Manager query returns solely items of type `XYZ_AdjReport`. If you specify an XPath query that results in a set of items of a different type, the business process will fail with an exception.

Extending and Aggregating the Service

With this business process, you have built a new service which implements a certain interface and performs a certain task. In the SOA programming model, it is straightforward to reuse this new service and aggregate it to create new services based on the existing ones. The SCA programming model in WebSphere Integration Developer allows you to export the new service in several different ways (e.g. as SCA service, web service or JMS service) and to use it within other modules.

6. Hints and Troubleshooting

6.1 Considerations 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 a 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.
- See section 6.3 for details on how to clean up data used by the sample process.

6.2 Starting the server and launching the administrative console

Option A: from within WebSphere Integration Developer

1. In the **Servers** view of WebSphere Integration Developer right-click the name of your server profile and select start. This switches to the console view that displays the output messages of the server startup.
2. Wait until the message `Server server1 open for e-business` shows up in the console view. Note that other messages may follow so it may not be the last one that shows up during the startup procedure.
3. Switch to the **Servers** view, right-click the name of your server profile and select **Run administrative console**.
4. If security is enabled you may see a security alert. Click OK to open the logon page with your browser. If your workspace preferences have **Internet > Web Browser** set to **Internal Web Browser**, the browser window opens in a view of your WebSphere Integration Developer. We recommend using an external browser which that uses a separate independent window.
5. To stop the server log out from the administrative console and ensure that no changes are pending by saving or discarding uncommitted changes, right-click the server profile in the **Servers** view and select **Stop**. The message `Server server1 is stopped.` signals successful completion of the step.

Option B: from the command line

1. Optionally clean up the server logs by deleting all files in the folder `<WPS_PROFILE_HOME>\logs\server1`.
2. Start the process server by running the command script `<WPS_PROFILE_HOME>\bin\startServer.bat` (hint:you may want to create a script that cleans up the logs and then starts a process to automatize the cleanup).
3. The startup is completed as soon as the message `Server server1 open for e-business` shows up. Note that other messages may follow so it may not be the last one that shows up during the startup procedure.
4. Open `<WPS_PROFILE_HOME>\logs\server1\SystemOut.log` with an editor and search for a line that says `Web Module adminconsole has been bound to admin_host[*:9060,*:9043]`. The first number shown between the square brackets is the `<ADMINPORT>` and the second one is the `<SECURED_ADMINPORT>` that needs to be used if security is enabled.
5. To start the administrative console, point your browser to `<http://<HOSTNAME>:<ADMINPORT>/ibm/console` (if security is not enabled) or `<https://<HOSTNAME>:<SECURED_ADMINPORT>/ibm/console` (if security is enabled).
6. To stop the server log out from the administrative console and ensure that no changes are pending by saving or discarding uncommitted changes, run `<WPS_PROFILE_HOME>\bin\stopServer.bat` and wait for the message `Server server1 is stopped.` which signals successful completion of the step.

6.3 Clean up the process environment

1. Stop the test server.
2. 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.
3. Delete all items of the result list.
4. Connect to `icmnlsdb` as `icmadmin` and drop the three tables `bpecontentevents, bpecollectionpoints, bpecollectionpointitemtypes`.
5. Restart the test server. This re-creates the three tables.

6.4 Install JRE update on WebSphere Process Server

With the just in time compiler enabled the server may terminate due to a mismatch between the Java Runtime Environment and the legacy DB2 JDBC driver. Disabling the just in time compiler is a quick but undesirable fix with significant performance impact. It is recommended to update the Java Runtime Environment to the latest interim fix or fixpack to get rid of this problem. By the time this documentation is published, this is Service Release 4 that can be downloaded from this support page. The required files and instructions are on the support site. Here are some hints that help to install the update:

1. Download the ZIP file of the update installer into a temporary folder and unzip it into the `<WPS_HOME>` folder.
2. Download the interim fix into `<WPS_HOME>\updateinstaller\maintenance`.
3. Run `<WPS_HOME>\updateinstaller\update.exe`.
4. Verify the information on the update installer pages. The default settings should be OK. Click **Next** to proceed through the pages and **Relaunch** to re-start the update installer with the updated JRE. When the update has been completed successfully, click **Finish** to close the update installer.
5. The Java Runtime Environment of the WebSphere Process Server has now been updated to the required fix level so the Just in Time Compiler can be re-enabled as follows:
   - Start `server1` and log on to the administrative console.
   - In the navigation panel select `Servers > Application Servers`, and click **server1**.
   - On the configuration page locate the section **Server Infrastructure**, expand **Java and Process Management**, and select **Process Definition**.
   - In the **Additional Properties** section of the **Process Definition** page select **Java Virtual Machine**.
   - Uncheck **Disable JIT** 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.
   - Log out from the administrative console and re-start the server.

6.5 Recreating the content viewer applet

The Quick Start Client uses a modified version of the DB2 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 DB2 Content Manager install directory and (re-)creates the JAR file for the modified viewer applet. This step should be performed whenever the code in `TViewerApplet.java` has been changed.

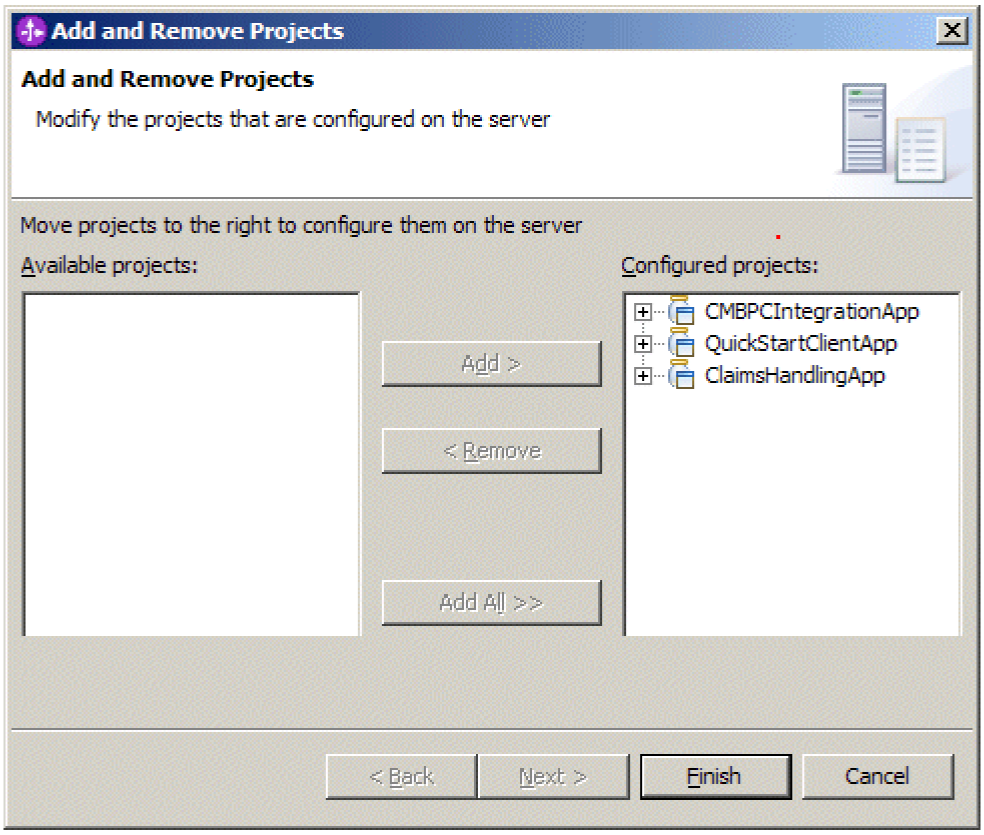
1. Switch to the **Physical Resources** view of the business integration perspective.
2. Expand **QuickStartClient > WebContent**.
3. Perform the following steps to import `cmbview81.jar` into this folder.
   a. From the workspace menu select **File > Import**. The 'Import' window opens.
   b. Select **File system** and click **Next >**.
   c. Click the **Browse** button to the right of the **From directory:** field, locate the folder `<IBMCMROOT>\lib`, and click **OK**.
   d. In the field to the right select `cmbview81.jar`.
   e. Enter `QuickStartClient/WebContent/appletViewer` in the **Into folder:** field.
   f. Click **Finish**.
4. Right-click `QuickStartClient/WebContent/appletViewer/cmbview81.jar` and select **Refresh**.
5. 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:
   - From the menu select **File > Export**.
   - Select **JAR file** and click **Next>** which switches to the 'JAR Package Specification' dialogue.
   - In the section **Select resources to export** select `ContentViewerApplet`.
   - 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**.
6. In the 'Package Explorer' view right-click `QuickStartClient/WebContent/appletViewer/appletViewer.jar` and select **Refresh**.

6.6 (Re-)deploying the applications on the test environment

This section explains the steps required to optionally clean up the work space and (re-)deploy the three applications on the test environment.

1. If any of the three applications `CMBPCIntegrationApp`, `QuickStartClientApp`, or `ClaimsHandlingApp` is currently deployed on the server, perform the following steps to undeploy them:
   1. right-click the server configuration in the **Servers** view, select **Add and remove projects**
   2. Click **Select All**, and click **Finish** to remove them.

    3. Wait for the progress indicator to signal completion of this step.
2. Optional: from time to time it might be useful to clean up the workspace to ensure all data in and across projects is consistent. To clean up the workspace proceed as follows:
    1. On the main menu select **Project > Clean**. Ensure **Clean all projects** is selected and click **OK**. Wait until the progress information signals completion of the step.
    2. Click the **Problems** tab to check for potential problems.
    3. If the **Problems** view displays an error `The project cannot be built until its prerequisite XXX is built. Cleaning and building all projects is recommended` (where XXX is a project name such as `CMBPCIntegration`) you can get rid of this error with the following trick: on the main menu select **Project > Build Automatically** which switches off the automatic build mode. Do the same again to switch the automatic build mode back on again. This re-builds the projects and removes the error message. Wait for the progress information that says **Building workspace** to complete.
3. Make sure no errors show up in the **Problems** view (warnings may be ignored).
4. Switch to the **Servers** view, right-click the server configuration, select **Add and remove projects**, click **Select All**, and click **Finish**.



5. Wait for the progress information to signal successful deployment of all three applications.
6. Inspect the console output (or `<WPS_PROFILE_HOME<\logs\server1\SystemOut.log` and `<WPS_PROFILE_HOME<\logs\server1\SystemErr.log`) for potential problems.
7. Optionally investigate the lines that start with >>. These contain output from the Integration Toolkit. The startup sequence should, for example, list the message `>> CMBPCIntegration services started` that signals successful initialization of the asynchronous bean. If Content Event Handling is used (based on attributes `WF_OnCreate` or `WF_OnDelete`, you may see messages of the type `>> Dropping trigger WP...` or `>> Adding trigger ... to column ... of table ...`. If Content Event Handling or the Collection Point mechanism is used the first time, the startup sequence will contain messages that indicate the creation of the corresponding tables (`BPECONTENTEVENTS`, `BPECOLLECTIONPOINTS`, and/or `BPECOLLECTIONPOINTITEMTYPES`. Potential warnings may savely be ignored.

6.7 Manually Starting and Stopping ClaimsHandlingApp

The enterprise application `ClaimsHandlingApp` can be started as follows:

1. In the WebSphere administrative console follow the links to **Applications > Enterprise Applications > ClaimsHandlingApp**.
2. Check the box left of the ClaimsHandlingApp application. Click the **Start** button. Wait until the status column displays a green arrow.

3. Follow the links to **Applications > Enterprise Applications > ClaimsHandlingApp > EJB Modules > ClaimsHandlingProcessEJB.jar > Business Processes**. Check if the **ClaimsHandlingProcess** template has been started. If not, click the **Start** button and wait until the progress information disappears.

To stop an enterprise application `ClaimsHandlingApp` perform the following steps:

1. Right-click the server configuration and select **Launch > BPC Explorer**
2. Log on as WebSphere administrator (e.g. `wasadmin` or `icmadmin`) and select **Process Instances > Administered by me**.
3. Select all available process instances and click terminate.
4. If any process instances are left: select them and click **Delete**.
5. Log off from the BPC Explorer.
6. Start the WebSphere administrative console and log in as administrator.
7. Follow the links to **Applications > Enterprise Applications > ClaimsHandlingApp > EJB Modules > ClaimsHandlingProcessEJB.jar > Business Processes**, select the **ClaimsHandlingProcess** template and click the **Stop** button. Wait until the progress information disappears.
8. Follow the links to **Applications > Enterprise Applications > ClaimsHandlingApp**.
9. Check the box left of the ClaimsHandlingProcess application. Click the **Stop** button. Wait until the status column displays a red cross.

6.8 (Re-) generating a JAR file for the DB2 Content Manager-based user registry

Perform the following steps to generate a (new) `icmuserregistry.jar` from your workspace:

1. In WebSphere Integration Developer select **File > Export** from the menu and select **JAR file**.
2. On the **JAR Package Specification** window select `CMConnectionManagement`
3. Enter `<WPS_HOME>\lib\icmuserregistry.jar` into the **Select the export destination** field.
4. The message `JAR export finished with warnings. See details for additional information` may show up where the details say `/CMConnectionManagement/META-INF/MANIFEST.MF was replaced by the generated MANIFEST.MF and is no longer in the JAR.`. This message can safely be ignored. Click `OK` to close the message window.

6.9 Re-activating the console output in WebSphere Integration Developer

With global security enabled, the console view of WebSphere Integration Developer may refuse to display log files. A solution for this problem is described in this Technote.

6.10 Avoiding conflicts when invoking CM WebServices from a process that also uses the Integration Toolkit

The XSD specification allows different data types to have the same QName. cmbwebservices.jar in `<IBMCMROOT>\lib` contains an instance of `cmbmessages.xsd` which shares QNames with definitions in `cmbmessages_modified.xsd` that has been imported from the Content Manager administration client. To ensure definitions with the same QName occur only once on the classpath, perform the following steps when using the Web Services in combination with the Integration Toolkit:

1. Start the administrative console and log on as `icmadmin`
2. In the navigation pane select **Servers > Application Servers > server1**
3. Locate the section **Server Infrastructure** to the right, expand **Java and Process Management**, and select **Process Definition**. In the **Additional Properties** section click **Java Virtual Machine**.
4. Click **Custom Properties** in the **Additional Properties** section to open the custom properties of the JVM.
5. Click `ws.ext.dirs` and replace `${IBMCMROOT}/lib;` with `${IBMCMROOT}/lib/cmbsdk81.jar;${IBMCMROOT}/lib/cmb81.jar;${IBMCMROOT}/lib/log4j-1.2.8.jar;`.
6. Click **OK** to update the property.
7. Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.
8. Log out from the administrative console and re-start the server.

The following steps are only needed if you plan to use the Quick Start Client:

1. In WebSphere Integration Developer open the deployment descriptor of the Quick Start Client as follows:
   - Switch to the J2EE view.
   - In the project view locate **Enterprise Applications > QuickStartClientApp > Deployment Descriptor: QuickStartClientApp**.

- Double-click `Deployment Descriptor: QuickStartClientApp` to open it with the editor.

2. Switch to the **Deployment** tab and scroll down until the **Shared Library** section appears at the lower right corner.
3. Expand the section and click **Add** to add a new library.
4. Enter `ICMLIB` into the **Name:** field.
5. Optional but recommended: enter `DB2 Content Manager libraries for use with the CM/WPS Integration Quick Start` into the **Description:** field.
6. Enter the following libraries into the **Class path:** field by clicking **Add**, entering the path, and clicking **OK**:
   - `${IBMCMROOT}/lib/cmbservlets81.jar`
   - `${IBMCMROOT}/lib/cmbtag81.jar`
   - `${IBMCMROOT}/lib/cmbview81.jar`
7. Click **OK** to store the definition.
8. Press CTRL-S to store the updated deployment descriptor and close the editor.

6.11 Re-enable credential caching for the Content Manager-based user registry

If the following sequence of messages shows up test server console periodically (approx. every second) this means that credential caching is disabled and the server has to go to DB2 Content Manager each time it needs to authenticate the user `icmadmin`. Perform the steps listed below to re-enable credential caching.

```
>> CM WAS User Registry - getUniqueUserId [...]
>> CM WAS User Registry - getUniqueGroupIds, user=ICMADMIN
>> CM WAS User Registry - check password for user icmadmin
Cleaning up CM connection...>> successful.
```

1. Launch the administrative console of the test server and log on as `icmadmin`.
2. In the navigation panel select **Security > Global Security**.
3. In the **General Properties** section set the **Active authentication mechanism** to `Simple WebSphere Authentication Mechanism (SWAM)` and click OK.
4. Click **Save** on top of the window to open the **Save to Master Configuration** window. Click **Save** to make the changes permanent.
5. Re-start the server.
6. Repeat the above listed steps to set the **Active authentication mechanism** back to `Lightweight Third Party Authentication`.

7. Further Reading

Product documentation

- WebSphere Business Process Management Version 6.0 information center
- WebSphere Process Server documentation
- WebSphere Integration Developer documentation
- DB2 Content Manager V8.3 Infocenter

Background information and documents on specific topics

- Technical Overview of WebSphere Process Server and WebSphere Integration Developer (IBM Redpaper REDP-4041)
- Content Manager Implementation and Migration Cookbook (IBM Redbook SG24-7051)

Product support Web sites

- WebSphere Integration Developer Support
- WebSphere Process Server 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