
Lab 1 Set Up the Process Enactment Environment

In this lab you will set up your environment for customizing processes. This involves installing client and server software for writing process descriptions and editing process definitions.

Process development should occur in a separate Rational Team Concert (RTC) web application from the regular production server. You can use the same Jazz Team Server for both applications. You will be creating many projects to incrementally develop and test your process definitions. RTC projects can't be deleted at this time, so developing them in a separate RTC application keeps the production environment cleaner. It's also good to develop and test processes in a separate RTC application so you don't have to worry about accidentally changing settings in the version of RTC that everyone else is using. You'll be free to experiment and create without impacting other team members.

Describing your process is done in Rational Method Composer (RMC).

You will install the following software in this lab:

- JTS and RTC servers used by the organization. For the purposes of the workshop, these represent the real-life JTS and RTC web applications your organization uses to manage projects, work items, etc. We'll refer to these as the "production" JTS and RTC servers in this workshop.
- An RTC server used for process development. This is a separate RTC application where you will develop your process templates.
- RMC.

When you're through with this lab, you will have learned:

- How to install all the software necessary to describe and define your process.
- How to set up a structure for editing process templates and process descriptions.

1.1 Download Files

- __1. Create a directory in the root folder of your local disk. E.g. [C:/PEW](#) for windows, or root/PEW for Linux. Create a sub-directory in PEW named Downloads. Download all files into this directory.
- __2. Download and install Firefox 10 ESR. <http://www.mozilla.org/en-US/firefox/organizations/all.html>
 - __a. Select the Firefox > Options and change the following settings:
 - __i. General tab
 - __a. Save files to root/PEW/Downloads
 - __ii. Content tab
 - __a. Turn off popup blocking. CLM applications sometimes use popups for logging in.
 - __iii. Advanced
 - __a. Select *Never check for updates*. As of this writing, Firefox 10 is the version supported by CLM, so you should keep it at that level.
- __3. Download CLM
 - __a. Go to <http://jazz.net> , click Downloads, and click the page to download the latest version of Rational Team Concert.
 - __b. Click *More download options*.
 - __c. In the Plain .zip section, subsection *Jazz Team Server and the CCM Application, and Trial licenses for Rational Team Concert* , download the zip file for the platform you're using.
- __4. Download Rational Method Composer.
 - __a. Download the [Rational Method Composer 7.5.2 free trial](#).

1.2 Set up Jazz Server

- __1. Install the “production” JTS and RTC applications.
 - __a. Create a folder root/PEW/IBM/JazzTeamServer and unzip the JTS zip file into this folder.
- __2. Set up hostnames URIs
 - __a. Set up fully qualified domain names.

- __i. Open the hosts file for editing as an administrator. In Windows right-click Notepad.exe and run it as an administrator. On Windows, the hosts file is located at C:\Windows\System32\drivers\etc\hosts. On Linux, /etc/hosts
- __ii. Add the following line to the bottom of the file: **127.0.0.1 clm.process.ws**
- __iii. Save and close the file.

What Did I Just Do?

By adding that line to the hosts file, you have told your networking software that any requests to a machine named `clm.process.ws` should be routed to the 127.0.0.1 address (which is your local machine).



If you were doing this in a REAL environment, you would have your systems administrators make an entry into the DNS tables (which are used to resolve hostnames) that would route the base of your selected Public URI to the machine hosting your Jazz infrastructure.

- __b. Configure Tomcat to serve applications on HTTP and HTTPS default ports:

We're doing this because...

We have decided that our public URI for CLM deployment is going to be `clm.process.ws`. We want it to be served with no reference to ports on the server to make user access easier.



This will also allow our installation to scale in the future and leverage middleware like Web Server or DNS virtual names. We won't be scaling to large topologies in this workshop, but it's a good practice to install this way when doing evaluation topologies. It will make future changes easier if your topology needs to evolve into a more robust environment.

- __i. Open the server configuration file `root/PEW/IBM/JazzTeamServer/server/tomcat/conf/server.xml` for editing.
- __ii. Look for the string '9443'
 - __a. Change all the occurrences in non-commented nodes of that *port* and *redirectPort* attribute value to **443**
- __iii. Perform another search looking for the string '9080'
 - __a. Change the occurrences in non-commented nodes of that *port* attribute value to **80**

**Information**

Just the Connector nodes for HTTP and HTTPS are required to be changed for CLM. However, you don't want redirections to a port you don't want to use in your deployment and which may be in use by other services.

The resulting nodes for HTTP and HTTPS will look like the following:

```
...
<Connector port="80" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="443" />
...
<Connector port="443"
    connectionTimeout="20000"
    maxHttpHeaderSize="8192"
    maxThreads="150"
    minSpareThreads="25"
    enableLookups="false"
    disableUploadTimeout="true"
    acceptCount="100"
    scheme="https"
    secure="true"
    clientAuth="false"
    keystoreFile="ibm-team-ssl.keystore"
    keystorePass="ibm-team"
    protocol="HTTP/1.1"
    SSLEnabled="true"
    sslProtocol="${jazz.connector.sslProtocol}"
    algorithm="${jazz.connector.algorithm}"
    URIEncoding="UTF-8" />
```

__i. Save your changes and exit.

__3. Add the second CCM application to the Tomcat server. This is the CCM application you will use for process development.

Two CCM applications in one Tomcat server

You are setting up your environment so two CCM applications (one for regular software development, one for process development) are associated with a single JTS and are serviced by one Tomcat application server. You can have as many CCM applications associated with a single JTS as you want, and a Tomcat server can service many web applications.



However, each web application must have a different name (also referred to as a context root). So you need to move your CCM application for process development into the Tomcat server and change its name. This requires moving some files and editing some configuration files.

For more information, see this [article on using multiple CCM applications in a single Tomcat server](#).

- ___a. Duplicate the directory `root/PEW/IBM/JazzTeamServer/server/conf/ccm` and call it `root/PEW/IBM/JazzTeamServer/server/conf/ccm-pew`. You can simply copy/paste the directory, then rename the new directory.

**ccm-pew?**

This application name refers to the CCM application you will use to develop, maintain, and deploy your processes.

- ___b. Duplicate `root/PEW/IBM/JazzTeamServer/server\tomcat\webapps\ccm.war` and call it `ccm-pew.war`.
- ___c. Open `root/PEW/IBM/JazzTeamServer/server\conf\ccm-PEW\teamserver.properties` in a text editor.
- ___d. Change directory references from `ccm` to `ccm-pew`
 - ___i. Locate all instances of the text `conf/ccm`. There will probably be 2.
 - ___ii. Change that text to `conf/ccm-pew` (leave the rest of the text on the line alone).
 - ___iii. Save and close `teamserver.properties`.
- ___e. Edit the `provision_profile.ini` files to point to the `ccm-pew` directory.
 - ___i. Navigate to `root/PEW/IBM/JazzTeamServer/server\conf\ccm-PEW\provision_profiles`.
 - ___ii. Locate 3 `.ini` files in this directory (`profile.ini`, `rtc-commons-profile.ini`, and `enterprise-update-site.ini`). Open all of them in a text editor.

- ___iii. In each file, change the text **file:ccm/sites** to **file:ccm-pew/sites**.
- ___iv. Save and close the files.
- ___4. Set up the CLM applications and JTS server
 - ___a. Startup the JTS server by running <root/PEW/IBM/JazzTeamServer/server/server.startup>.
 - ___i. Open a browser and run setup by navigating to <https://clm.process.ws/jts/setup>. Ignore any security warnings and add a security exception if asked.
 - ___ii. Login to JTS with the login/password ADMIN/ADMIN.
 - ___iii. Select *Express Setup*, then **Next**.
 - ___iv. In Configure Public URI, assure <https://clm.process.ws/jts> is the Public URI.
 - ___v. Select *I understand that once the Public URI is set, it cannot be modified*, then select **Next**.
 - ___vi. On the Create User page, create the PEW admin user then select **Next**:
 - ___a. User ID: *pewadmin*
 - ___b. Name: *pewadmin*
 - ___c. Password: *pewadmin*
 - ___d. Email: *pewadmin@bogus.ws*



Error During Setup

If you receive an error during setup, it's most likely because the files in `server/conf/ccm-pew` were not edited correctly. Return to that section of the lab and make sure you have edited the files correctly. Shutdown (`server.shutdown`) and re-start (`server.startup`) the server, then run setup again.

You will probably need to log in as `pewadmin/pewadmin`, and you will be required to run Custom Setup. You can select the defaults on any pages that are not otherwise described in these instructions.

- ___vii. Select **Next** when *Express Setup* is complete.
- ___viii. On the Assign Licenses page:
 - ___a. Under *Rational Team Concert*, next to *Rational Team Concert – Developer*, select **Activate Trial**.

- __ix. Select **Finish**.
- __x. On the Server Administration page, select the link for *Create Users* and enter the following values then select **Save**:
 - __a. Username: *Jim*,
 - __b. User ID: *jim*
 - __c. email address *jim@bogus.ws*
 - __d. Repository permissions: **JazzAdmins** (you can leave the default selected *JazzUsers* too).
 - __e. CALs: **Rational Team Concert – Developer**.

You have set up your Jazz Team Server with two CCM applications (ccm and ccm-pew).

1.3 Set Up Rational Method Composer (RMC)

__1. Install RMC into RTC

__a. Download the [IBM Rational Method Composer 7.5.2 trial](#).



There's No Such Thing as a Free Lunch

You may have to fill out a short survey in order to download the RMC 7.5.2 trial.

__b. Launch the RMC installation file you downloaded from developerWorks.



Shell Sharing RMC and RTC: 32- or 64-bit?

RMC 7.5.2 is a 32-bit application. On Windows, it can be run in the same shell as the 32-bit or 64-bit version of RTC.

As of this writing, if you shell-share RMC 7.5.2 on Linux with RTC, then RMC **must** run in a 32-bit version of RTC. If you try to install RMC on Linux when you only have a 64-bit version of RTC, you will be forced to create a new package group. This new group will install a separate installation of Eclipse with RMC. It will not share an Eclipse shell with RTC.

__c. Make sure RMC is selected, and select **Next**.

__d. If you accept the license agreement, select **Next**.

__e. If you are prompted with a poll, fill the required values and select **Next**.

__f. By default RMC will install to a new package group. (Optional: To shell share with a Rational Team Concert client, select **Use the existing package group**, and select the package group for *IBM Rational Team Concert Client*.) Select **Next**.

__g. Verify the packages to install are correct, then select **Next**.

__h. Select **Install** on the final screen.

__i. When the installation is finished, close *Installation Manager*.

__2. Update RMC

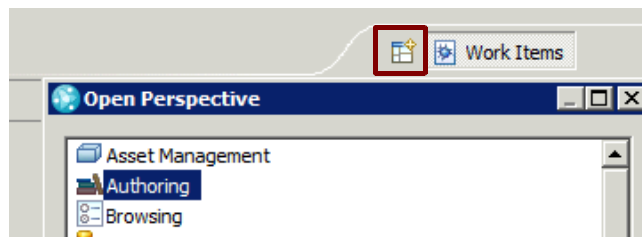
__a. <Update to RMC 7.5.2.2 – TBD>

- ___b. If using 64 bit Linux, install xulRunner per instructions here:
<http://www-01.ibm.com/support/docview.wss?uid=swg27016986>

___3. Set up RMC

- ___a. Windows: Set up a new workspace for RMC.
 - ___i. Click **Start > All Programs > Rational Method Composer**
 - ___ii. Right click on **Method Composer**
 - ___iii. Click *copy*
 - ___iv. Right click on the desktop and click *paste shortcut*.
 - ___v. Right click on the shortcut and click *properties*
 - ___vi. Append “ -data C:\PEW\Workspaces\Lab1” (without quotes, and including an initial space) to the *target*.
- ___b. Linux: Windows: Set up a new workspace for RMC.
 - ___i. Open a terminal window, and enter:
`gnome-desktop-item-edit ~/.local/share/applications -create-new`
 - ___ii. Set the command to:
 - ___iii. `<rmc install location>/rmc/rmc -data PEW/Workspaces/Lab1`
- ___c. Launch RMC using the shortcut
- ___d. Skip the option to copy the library
- ___e. Close the *Welcome* view.
- ___f. Set up preferences for RMC.
 - ___i. Select **Window > Preferences**
 - ___ii. In **Method > Authoring**, select the following:
 - ___a. *Use new Extends semantics*
 - ___b. *Link name with presentation name for new method element.*
 - ___c. Leave all other settings as-is.
 - ___iii. In **Method > Authoring > Process Editor**, select the flowing:

- ___a. *Use auto synchronization for all processes.*
- ___b. *Leave all other settings as-is.*
- ___iv. In **Method > Publishing/Browsing**, select the following:
 - ___a. *Include method content in descriptor pages*
 - ___b. *Leave all other settings as-is.*
- ___v. In **Method > Publishing/Browsing > Activity Diagram**, select the following:
 - ___a. *Select Publish activity diagrams for unmodified activity extensions*
 - ___b. *Publish activity detail diagrams that have not been created in process editor.*
- ___vi. In **Method > Publishing/Browsing > Look and Feel**, select the following:
 - ___a. *Skin: RMC Compact.*
- ___vii. In **Team>Jazz Source Control> Check-in Policies**, select the following:
 - ___a. *Auto check-in local changes*
 - ___b. *Perform check-in whenever a resource is modified*
 - ___c. *Leave all other settings as-is.*
- ___viii. Select **OK**.
- ___g. Download PEW library (should be available from the same location as this document).
 - ___i. Download to root/PEW/Downloads
 - ___ii. Unzip lib.PEW.zip to root/PEW/Downloads/lib.PEW
 - ___iii. In RMC, switch to the Authoring perspective.



- ___iv. Select **File>Open>Method Library**.
- ___v. Select *Open method library from workspace*, then **Finish**.

- __vi. Set the library to auto-synchronized.

Click *Window > Preferences > Method > Authoring > Process Editor*. Select the option *Use auto synchronization for all processes*.

__4.

Auto-synchronized Libraries



RMC auto-synchronization is a feature that automatically keeps task descriptors in-sync with their corresponding tasks. This is a library-wide setting and libraries that auto-sync can't be imported into manually synced libraries.

You cannot import non-autosynchronized library plug-ins into a auto-synchronized library.

If you see an error that says *importing an auto-synchronized library to a synchronization required library is not supported*, this means that you did not set the preference setting described in the previous step.

__5.

Use Workspace Libraries Instead of Static Libraries




An RMC Workspace Library stores all the content of your process inside your Eclipse workspace. This makes using SCM much more convenient.

Workspace libraries also make it easier to work on content with other people. Each person can load the RMC plugins they want, and each person's library is managed locally in their Eclipse workspace.

In a static library, the library is identical for everyone so everyone must load the same plugins and use the same structure. For multi-user RMC environments, it is almost always preferable to use workspace libraries.

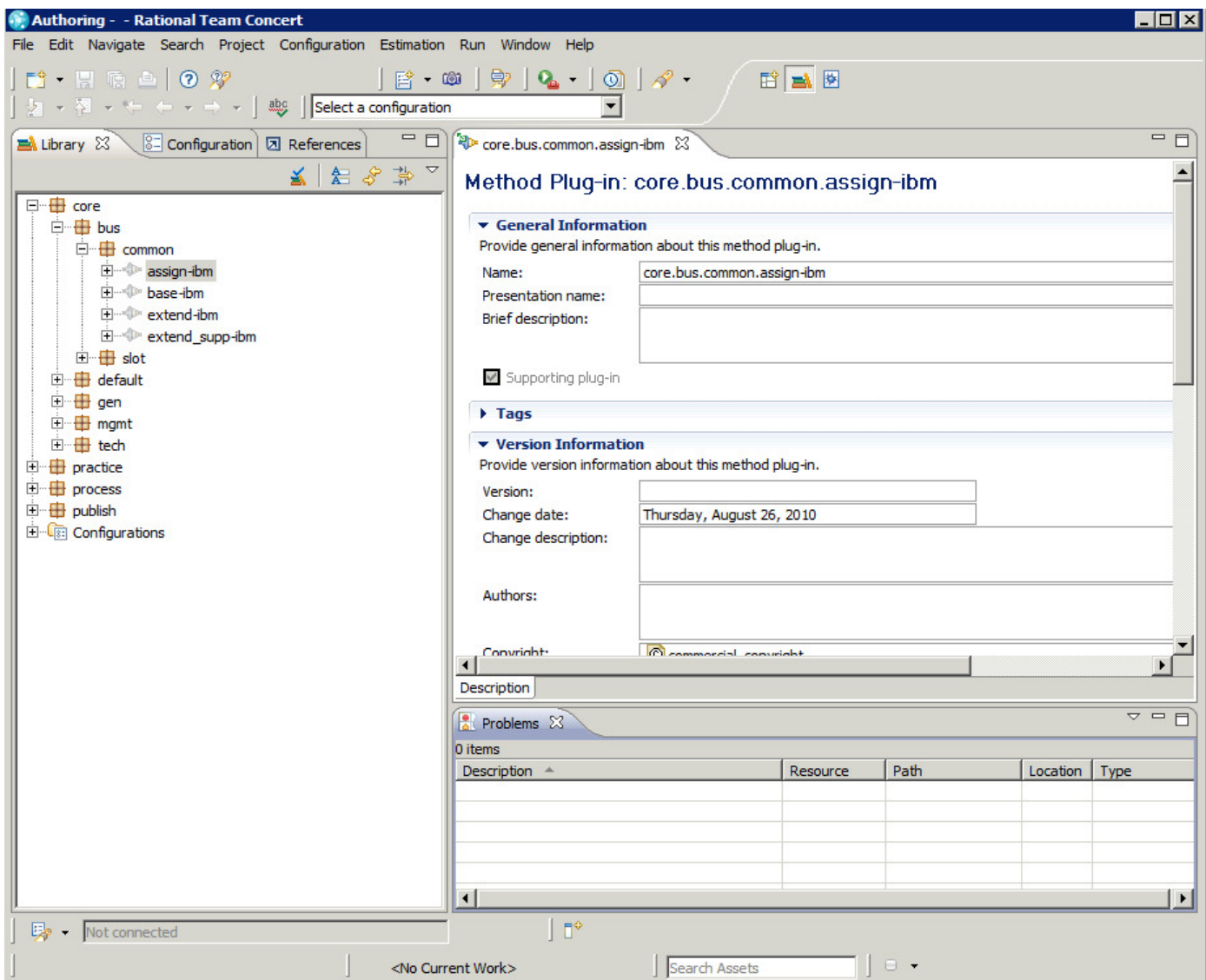
- __i. Import the PEW Library into the workspace
- __a. Select **File > Import > Existing projects into workspace**. Select **Next**.
 - __b. Browse to *root/PEW/Downloads/lib.PEW* and select **OK**. Select **Next**.
 - __c. Make sure all of the elements are selected, then select **Finish**

Don't Modify the IBM provided Library

- __5. After you import the library, open any plug-in. You'll notice there's a flag that indicates the plugin is locked. This means RMC will not allow you to modify the contents of the plugin.
- __6.
- __7. This is by design. The base libraries that ship with RMC should not be modified. All changes, additions, and content removal is done using variability in RMC.
- __8.
-  __9. In other words, to change something you'll create a new plug-in, add new elements to that plug-in, and apply those new elements to the existing elements using variability. This will add, change and remove content in the published process.
- __10.
- __11. This is a feature of SPEM, which RMC uses as the metamodel for process descriptions. The result is that you only need to put the plugins that you create into SCM. The original libraries will be the same for everyone, so they don't need to be maintained in SCM.
- __12.
- __13. This also makes it easier to upgrade method libraries. When you get a new version, just replace the old library with the new one. Variability will maintain your changes.

__b. Customize your Authoring perspective.

- __i. You can arrange the views of any perspective in any way you want. Here's one arrangement that saves space and maximizes the size of the views you'll use the most.



- __c. Create connections to the process and software development repositories
 - __i. Switch to the *Jazz Administration* perspective.
 - __ii. Select **Create a Repository Connection** in the *Team Organization* view. Enter the following information, and select Finish.
 - __a. URI: *https://clm.process.ws/ccm-pew*
 - __b. Name: *ProcessDev*
 - __c. UserID/Password: *jim/jim*
 - __iii. If you receive certificate warnings, accept the certificates permanently.

- ___iv. Create another repository connection by repeating the steps above, except for the following:
 - ___a. URI: *https://clm.process.ws/ccm*
 - ___b. Name: *SoftwareDev*
- ___v. Switch to the *Team Artifacts* view and you'll see the repository connections.

1.4 Set up the Process Development environment

- ___1. Create a process development project based on Scrum. *** Needs to be rewritten based on the web client ***
 - ___a. In the *Team Organization* view select **Create a Project Area**.
 - ___b. Select *Jim@ProcessDev*, then select **Next**.
 - ___c. Name: *Acme Process Development*, then select **Next**.
 - ___d. Select **Deploy Templates**. This is a one-time action after you set up CLM.
 - ___e. After the templates are loaded, select the **Scrum** process.
 - ___f. Select *Automatically initialize the Project Area on Finish as specified in the process template*. Select **Finish**.

When to Delay Initialization



You might want to delay initialization of the project if you're going to change any of the follow-up actions that are run during initialization.

For example, you may want to add or remove work item templates that automatically generate work items on project initialization.

- ___g. Add Jim as a member of the project and assign process roles.
 - ___i. In the *Overview* tab, select the **Members** area of the *Acme Process Development* window. Select **Add**.
 - ___ii. Enter *Jim* in the search bar, then select **Search**.
 - ___iii. *Jim* appears as a matching user. Select the **Select** button. Then select **Next**.
 - ___iv. Add *Team Member* as an *Assigned Role*, then add *Scrum Master*. Select **Finish**.

Order of Process Roles

Note that in the Process Roles section for “Jim”, Scrum Master is listed first.

The order in which roles are assigned in a process area (project area or team area), is important for RTC when there are preconditions or follow-up actions configured for an operation. These actions are called "operation behavior" and RTC uses the order of roles assigned to a user to decide which operation's behaviors are applicable to a user in the context of an operation (like saving a work item). The computation of roles begins from the iteration governing the operation, then the process area up through the hierarchy: these roles are checked in order of precedence looking for just one matching for the operation.



For example, for the configuration you have just performed, if you had configured the project area with preconditions for Team Member role and for Scrum Master role for the "Save Work Item" operation; whenever the user "jim" saves a work item in the context of the project area the precondition to be fired would be just the one for the Scrum Master. See [Process Behavior Lookup](#) for more information.

This is a bit different when it comes to permissions to perform an action: the calculation of assigned roles for the user is the same as for process behavior lookup, but for permissions the user will be granted based on the sum of the permissions for all the roles assigned in the context of the operation. See [Process Permission Lookup](#) for more information.

- __h. Create a pre-condition that ensures local workspaces have the latest material before they deliver change sets (see the screen shot at the end of this step for an illustration).

Avoid Merging RMC Content



In method authoring it's important to make sure you're using the latest material from other people so you don't step on their changes. Merging in RMC is difficult, so this pre-condition helps everyone grab the latest content before they check-in their own content, reducing the chance that someone will overwrite material or require manual merging.

- __i. Select the Process Configuration tab.

- __ii. Select Team Configuration > Operation Behavior.
- __iii. In the *Operations* box, scroll down to the *Source Control* category.
- __iv. Next to Deliver (server), click in the Everyone cell.
- __v. Select the checkbox *Preconditions and follow-up actions are configured for this operation*.
- __vi. Select the **Add** button next to the *Preconditions* box.
- __vii. Select *Require workspaces to be caught up before delivery*, then select **OK**.
*** I could not find this condition***
- __viii. Select **Save** in the project window. Your view should look like the following:

Configuration

- Roles
- Project Configuration
- Team Configuration
- Permissions
- Operation Behavior**
- Event Handling (unconfigured)
- Iteration Types
- Timelines

Operation Behavior

Select a cell in the table below to configure the preconditions and follow-up actions for the corresponding operation.

Preconditions are checked before running an operation; follow-up actions are executed after. An operation's preconditions and follow-up actions can be configured differently for each role. Note that operation configurations completely replace each other. The process runtime will choose the most appropriate operation configuration for the logged-in user and will execute the preconditions and follow-up actions defined in that configuration.

Operations	Everyone...	Product ...	Scrum M...	Team M
Display Report (server)				
Manage Report Folder (server)				
Source Control				
Deliver (client)				
Deliver (server)				
Deliver Phase 2 (server)				
Modify Component (server)				
Save Change Set Links and Comments (server)				
Work Items				
Delete Work Item (server)				
Save Work Item (server)				

The deliver operation is performed when changes or baselines are delivered from a workspace to a stream.

☒ Preconditions and follow-up actions are configured for this operation

☐ Final (ignore customization of this operation in child areas)

Preconditions (5 available):

Require Workspaces to Be Caught Up Before Delivery

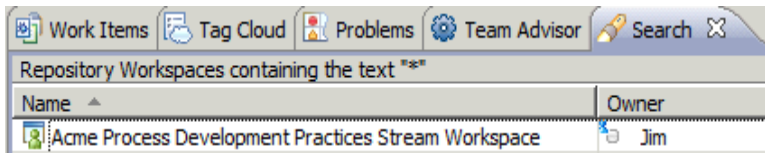
Add...

Name: Require Workspaces to Be Caught Up Before Delivery

Description: Only allow delivery from workspaces that are caught up with are delivering to.

- __2. Set up the Process Development workspace
 - __a. Switch to the *Team Artifacts* view.
 - __b. Rename the workspace.

- ___i. Right-click on **My Repository Workspaces** and select **Search for Repository Workspaces**.
- ___ii. Select *ProcessDev* as the Repository to search, enter * as the search pattern, then select **Search**.
- ___iii. The *Acme Process Development Stream Workspace* appears in the Search view. Double-click on the **Acme Process Development Stream Workspace**.



- ___iv. Change the name to *Acme Process Development*. Select **Save**.
- ___c. Create and rename components in the workspace.
- ___i. Right-click on the component called *Acme Process Development Default Component*.
 - ___ii. Select **Rename**.
 - ___iii. Rename the component *Process Extensions*. This is where you will keep the RMC process descriptions you create and modify. Select **OK**.
 - ___iv. Open the *Acme Process Development Stream Workspace*. Beside **Components** click **New...**
 - ___v. In the component name dialog, enter the name *Process Templates*. This is where you will keep the RTC process templates you'll create and modify. Select **OK**.
 - ___vi. Select **Save**.
 - ___vii. Right-click the Process Templates component and select **Change Owner**.
 - ___viii. Select *Acme Process Development* as the owner, then select **OK**.
- ___3. Add the *Scrum/Agile ALM(local practices)* process template
- ___a. Download the *Scrum/Agile ALM(local practices)* process template from the same location as this lab (file named *ibm.aalm.localpractices.r3.zip*)
 - ___b. In the browser, go to: <https://clm.process.ws/ccm-pew/admin>
 - ___c. Click on *Templates*. Click *Import Template*. Browse to select the downloaded process template.
- ___4. Add a component for SCM of process templates.

- ___a. Back in RMC, select **File > New > Other > General > Project**. Then select **Next**.
- ___b. Project name: *Acme Templates*. Use the default location, then select **Finish**.
- ___c. Share the Acme Templates project:
 - ___i. Select the *Navigator* view (add the view if it's not visible via *Window>Show View>Other>General>Navigator*).
 - ___ii. Right-click the *Acme Templates* folder and select *Refresh* to see the *openup.process.acme.ws* directory. Right-click on *Acme Templates* again and select **Team > Share Project**.
 - ___iii. In the *Share Project* window, select *Jazz Source Control*, then **Next**.
 - ___iv. Make sure the *ProcessDev* repository is selected. Select the option *Select a component or folder in an existing repository workspace*. Select *Acme Process Development > Process Templates*. Then select **Finish**.

Pending Changes



If you look in the Pending Changes view, you can see the files you've created. This indicates that they've been noticed, but not yet delivered to the stream. Recall in Lab 1 that you set an option to check-in files whenever they are modified (saved). So as soon as you added these files to SCM, the changes were checked-in to your server-based workspace.

The plug-in, configuration, and project files are safe in your server-based workspace. If anything happens to your local copy you'll still have a copy on the server in your workspace.

- ___d. Deliver the outgoing changes in the *Pending Changes* view.

1.5 Summary

Congratulations! You've set up your environment to enact your process by creating a project and adding preconditions. You now have the following in your process development environment:

- A simulated production environment with a JTS and associated CCM application. This is where you'll import your new process templates and descriptions to simulate how a real project would make use of them.
- A CCM application (ccm-pew) you'll use as a process development platform. This application is associated with the same JTS as the production CCM application.
- Rational Method Composer installed and preferences set.
- An RMC workspace library that contains the *PEW Library*. This library is read-only and will serve as a basis for your process development.
- A project, *Acme Process Development*, where you can manage the development of your processes and method content.
- An RTC workspace, *Acme Process Development Stream Workspace*, for editing RMC process content and modifying RTC process templates. This workspace flows changes to the *Acme Process Development Practices Stream*, which holds all process development components.

In the next lab you'll see how this environment is part of the cycle of creating process descriptions, enacting those processes in RTC, and harvesting changes back into the main RMC library.