Chapter 4
Administering RuleSets

A RuleSet is a container or an organizational construct that stores the rules that define all or part of a Process Commander application. Every instance of every rule type belongs to a RuleSet. Creating and managing RuleSets and RuleSet versions is a fundamental aspect of building a Process Commander application.

Usually RuleSets are created by the developers, either manually or by using the Application Accelerator. One RuleSet is created during the installation process so the development team can begin to work on the system right away. As a system administrator, you might also need to create a RuleSet, increment a RuleSet version, or install a product, patch, or RuleSet archive.

This chapter contains following sections:

- Overview of RuleSet and Application Rules
- Creating RuleSets and Application Rules
- Migrating Rules and Data
- Rule Maintenance
Overview of RuleSet and Application Rules

Application rules define a group of RuleSets as an application within the Process Commander system and orders the RuleSets according to their dependencies on each other. You assign application rules to access groups, which are then assigned to operator IDs. A user is able to create and/or use only those rules that belong to the RuleSets specified in the application rule in his or her access group.

RuleSets and Versions

Planning and creating RuleSets takes place early in the design of your Process Commander applications. Every RuleSet has a name and version to help you identify and manage its contents. The term RuleSet sometimes refers to the name of the RuleSet rule and sometimes refers to the group of rules that belong to the RuleSet.

The PegaRULES engine uses a combination of the rule type, purpose, RuleSet name and version, class structure, and security model to determine the correct business rule to apply in each situation as work is processed. This process is called rule resolution.

RuleSet versions use a numbering convention based on three dash-delimited numbers. For example: YourCoLoan:01-03-05. The first number, 01, represents the version; the second number, 03, represents a minor release within the 01 version; and the third number, 05, represents the interim release, or patch, within the 01 version.

RuleSet Dependencies

RuleSets are not complete by themselves; they must depend on other RuleSets, called prerequisite RuleSets, to provide additional rules needed for processing. For example, the organizational RuleSet that was generated during the installation process has the Pega-ProCom RuleSet specified as its prerequisite RuleSet.

Typically, you use one RuleSet to contain rules that apply to all applications and all users in an organization. Then, if you need additional RuleSets for your applications, they specify the main RuleSet as their prerequisite. As Figure 4-1 illustrates, when several RuleSets make up an application, only one RuleSet – the RuleSet at the
bottom of the hierarchy – should specify Pega-ProCom as the prerequisite RuleSet. The RuleSets higher up in the hierarchy depend on the next one down.

Figure 4-1. RuleSet Hierarchy

RuleSet Naming Conventions

The following naming conventions represent best practice:

■ RuleSet names cannot exceed 64 characters. For ease of management, use names that are short and easy to remember.

■ Always begin your RuleSet name with something that uniquely identifies your company and the business purpose. This convention also prevents potential RuleSet collisions at a customer site.

■ Don’t use Pega or Pega- as a prefix for your RuleSet names. This usage is restricted for Pegasystems use and can cause unexpected behavior.

■ For legibility, use single-word names with significant letters in uppercase (also referred to as title case). For example, CandyOrders is easier to read than candyorders.

■ It is best not to use special characters such as dashes (which signify class hierarchy levels), underscores, plus signs, or quotation marks.
Use names that make sense, avoiding obscure acronyms. The name should clearly describe the RuleSet. For example, QuadStateMortgage is better than QSGMLS (as an acronym for Quad State Group Mortgage Loan System).

For a test RuleSet, use a name that contains Test or Sample.

Rule Management

When a RuleSet has rule management enabled, users must check out a rule to modify it. When they check out a rule, a copy is placed in a private RuleSet (identified by the user’s operator ID) where they can make and test changes without affecting the original RuleSet.

While a user has the rule checked out, the rule in the original RuleSet is locked so others cannot access it until the local copy is checked back in or deleted. When users finish with a checked out rule, they must check it back in to make it available to others.

If you subject a RuleSet to rule management, the Check Out and Check In icons appear in the menu bar of the rule forms of all rules in the RuleSet.

Packaging Applications with Application Rules

You use an application rule to define and order the RuleSets that make up your application. One application rule can be dependent on (built on) another application rule.

After an application has been developed, you can move it to a test system to test the implementation, and, from there, to a production system where it is put into use. Process Commander provides several rule types and utilities to help you package an application, export it as a ZIP archive, and then import it into the other system. (Details on this process are provided later in this chapter.)
Because application rules serve as a container for RuleSets, they do not belong to RuleSets themselves. Instead, you specify which of the RuleSets the application rule should be associated with. That way, when you create a ZIP archive that includes the associated RuleSet, the application rule is included with the other rules in that RuleSet.

**Production RuleSets**

When an application is put into production, the RuleSets that make up the application are locked. A *production* RuleSet is an unlocked RuleSet used by a specific group of users if modifications need to be made to an application while it is running on a production system. These RuleSets should be used for small changes with little impact to the system and that apply to a small subset of users with special needs beyond the definition of the main application.

For example, perhaps a group of managers must be able to modify the selection criteria for reports. Or perhaps a developer wants to preview a proposed UI or functional change before it is implemented system-wide (in one of the main application RuleSets). In either case, you could use a production RuleSet.

You assign production RuleSets to the access groups of the appropriate users. Then those users can create or modify rules in that RuleSet if they need to update or override the behavior of a rule in one of the locked application RuleSets.
Creating RuleSets and Application Rules

This section describes how to create RuleSet, RuleSet version, and application rules. When creating rulesets, you can use the “quick create” method that generates both a RuleSet and a RuleSet version rule, or you can create RuleSet and RuleSet version rules separately.

Creating a New RuleSet and Version Together

To create a new RuleSet and version, you need to know its name and the prerequisite RuleSet.

Complete the following these steps:

1. From the menu, choose Application > New > Ruleset. The New form appears (Figure 4-2).

   ![Figure 4-2. New RuleSet and Version Form](image)

2. In the Name field, enter the name of your RuleSet. (This field is required.) The name cannot exceed 64 characters. Version 01-01-01 appears automatically in the Version field.
3. In the Prerequisite field, enter the name and version of the RuleSet your RuleSet depends on. For your first RuleSet, this will likely be the highest version of Pega-ProCom; for example, Pega-ProCom:05-01-08.

4. In the Description field, enter a short, meaningful description of the RuleSet.

5. To require users to check out rules from this RuleSet to work on them and then check them back in when they are finished, choose the Require Checkout option.

6. Click Create. Process Commander creates both a RuleSet and a RuleSet version rule.

Creating a New RuleSet Rule Individually

To create new RuleSet rules individually, complete the following steps:

1. From the Rules by Type explorer, choose SysAdmin > RuleSet. A list of RuleSets appears.

2. Click the New button.

3. In the New form, enter the name of the RuleSet and click Create. The RuleSet Name form appears (Figure 4-3).
4. On the Security tab, specify whether you want to use the rule management feature and whether you want to impose additional security for the RuleSet. You can specify that users must enter a password before they can perform the following tasks:
   - Add a RuleSet version for the RuleSet
   - Update a version for the RuleSet
   - Update the name of the RuleSet

5. You can also implement an approval flow that further restricts whether rules that belong to the RuleSet can be checked in. For more information, see the Application Developer Help.

6. Complete the History tab with a full description of the rule and information on its use.

7. Click Save.
Creating a New RuleSet Version Individually

Every RuleSet has at least one version, numbered 01-01-01. The best practice for version numbering is as follows:

- Create new versions to contain enhancements, bug fixes, and other changes.
- For bug fixes or minor changes, increment the last two digits of the version number.
- For enhancements or medium-level changes, increment the middle two digits of the version number.
- For major releases, increment the first two digits of the version number.
- You can use any increment you choose. For example, you could create 01-01-03 without first creating 01-01-02.

To create a new RuleSet version individually, you need to know which RuleSet is being incremented and what the new version is. Then, complete the following steps:

1. From the Rules by Type explorer, choose SysAdmin > RuleSet Version. A list appears.

2. Click New.

3. In the New form, specify the RuleSet name and the version number. Then click Create. The RuleSet Version form appears (Figure 4-4).
4. On the Security tab, under the Requires RuleSet and Versions section, specify the prerequisite RuleSet version.

5. Leave the Locking and Password Values sections blank until you are ready to put the RuleSet into production.

6. Complete the History tab with a full description of the rule and information on its use.

7. Click Save.
Creating Application Rules

After the appropriate RuleSets exist, create the application rules that specify the collection of RuleSet versions that make up your applications. When creating the access groups that grant users their access rights in Process Commander, you specify the appropriate application rule, which grants them access to the rules in the application RuleSets.

To create an application rule, complete the following steps:

1. From the Rules by Type explorer, choose Security > Application. A list of application rules appears.

2. Click New.

3. In the New form, specify the name and version of the application and click Create. The Application rule form appears.

4. On the Definition tab (Figure 4-5), list the application RuleSets according to their dependencies. You can also specify a prerequisite application rule, if necessary.
5. Leave the Locking fields on the Publish tab blank until you are ready to put the application into production. Then, you specify whether the application rule is to be locked so no one can change the RuleSets or the order of the RuleSets unless they provide a password.

6. On the History tab, provide information for the Full Description and Usage fields.

7. Click Save.
Migrating Rules and Data

When an application is ready for testing, move it to a test system, and then to a production system. You use the following tools and options to move the rules and data instances that make up an application from one Process Commander system to another system:

- Product rules and patch rules. These rules create ZIP files that contain the rules from specified RuleSets, plus any data instances that are identified by queries or when rules.
- The Export Rules/Data wizard. This wizard can create ZIP files from product and patch rules (ZIP files that contain both rules and data), or it can create ZIP files from RuleSets that contain rules only.
- The Import Rules/Data wizard. This wizard uploads and imports ZIP files that contain rules and/or data.

A product or patch rule ZIP archive contains the following elements:

- All the rules from the RuleSets specified, including the product or patch rule, the RuleSet rules, the RuleSet version rules, and the application rules associated with the RuleSets. The rules and data are XML documents in Unicode characters.
- All the data instances identified by the queries or when rules (Rule-Obj-When) used by the product or patch rule.
- (Optional) An installation activity and ReadMe file.
- The pegamove.xml file, a configuration file that contains installation information.

A RuleSet ZIP archive contains the following elements:

- All the rules (in XML format) from the RuleSet specified, including the RuleSet rule, the RuleSet version rule, and the application rule associated with the RuleSet (if any). The rules are XML documents in Unicode characters.
- The pegamove.xml file.
When you create a ZIP archive of rules or rules and data, the ZIP file is stored in the Process Commander ServiceExport directory and a link to the file is displayed in your browser. You can download the ZIP file to your workstation and then upload it to the destination Process Commander system. Note that during upgrades or other system maintenance, the ZIP files in the ServiceExport directory may be deleted.

When you use the Export Rules/Data wizard, the wizard creates a log file to store status and error messages generated while it ran. These files are stored as an instance of the Log-PegaRULESMove class.

**Before You Begin**

Before you create a ZIP archive of rules or rules and data to move to another Process Commander system, consider the following guidelines:

- You must have administrator privileges on both the source and destination systems.
- It is best if the versions of Process Commander are the same on both systems. The version of Process Commander on the destination must be the same as that of the source, or higher. For example, you cannot move a RuleSet created in version 05-01-06 to a system based on version 05-01-01.
- Determine the RuleSet prerequisites of the RuleSet(s) you want to move. Determine whether the dependencies are already present on the destination system or whether you must move them too, and if so, be sure that you move them in the order that matches the dependency relationship.
- Verify that all rules are checked into the RuleSet you are moving. You cannot export rules that are currently checked out by a user. To examine a RuleSet for checked-out rules, choose View > Rules > Check Outs from the menu. If rules are checked out, talk to the developers who have them checked out.
- After you create the rule but before you generate the archive, lock the RuleSets and the application rules to prevent unexpected changes. For RuleSet versions, the locking fields are on the Security tab. For application rules, the locking fields are on the Publish tab.
Creating a Product Rule

A product rule identifies all the RuleSets, versions, data objects, and other parts of an application, such as an organizational hierarchy, workbaskets, access groups, and operators. You use this rule type to create a product archive with an associated installation activity that you can deliver to another Process Commander system.

Gather Information

Before you can define a product rule, you must determine the following information:

- The intended name and version of the product — The name must start with a letter and contain only alphanumeric characters. The version must include only alphanumeric characters, periods (.), or hyphens (-), but need not match a RuleSet version.

- The RuleSets and versions to include.

- The data instances to include — For an application to run correctly on the destination system, you must include instances of at least the following data objects:
  - Database table names (see Chapter 5, “Managing the Data”)
  - Class groups (see Chapter 5)
  - Access groups (see Chapter 6, “Configuring Access Rights”)
  - Operator IDs (see Chapter 6)
  - Organization (see Chapter 3, “Setting Up the System”)
  - Divisions (see Chapter 3)
  - Organization units (see Chapter 3)
  - Workbaskets (see Chapter 3)
  - Work groups (see Chapter 3)

Carefully consider any dependencies among data instances or between data instances and your rules. During uploading, data instances that are already present are not overwritten unless you choose to do so.
Create When Rules That Identify Data Instances to Include

The product rule form enables you to identify the data instances that should be included in the ZIP either directly by each individual instance or by using When rules (Rule-Obj-When) that identify them. A simple way to accomplish this is to create one When rule that applies to the Data-base class and that evaluates for the property value of the pxInsName property for your application, sometimes called the “visible key.”

The pxInsName property of a rule or class is defined in the top-level class of your application so it is present in every class and every rule that applies to that class or inherits from it. For example, assume that the top-level class of your application is named MyCo-. You could create a When rule that evaluates for the condition in which pxInsName is MyCo, as shown in Figure 4-6.

![Figure 4-6. When Rule for a Product Archive](image)

The entire string from the Value field is:

```
 @(Pega-RULES:String).contains(.pxInsName, MYCO)
```

To continue with this example, you could use this same When rule for each data class that has instances you want to include in your product rule (Figure 4-7).
After you collect the appropriate information and create the needed When rules, complete the following steps:

1. From the Rules by Type explorer, choose SysAdmin > Product.

2. Click New. The New form appears.

3. Enter the name and version of the product. The .zip extension is not required. Choose your RuleSet name and version.

4. Click Create. The Product form appears.

5. On the Contents tab, specify the RuleSets and versions.

6. (Optional) In the Data to Include section, choose or specify the data class of the instance you want to include, and then choose the appropriate When rule. Include a note that describes the data instance that will be included.

7. (Optional) In the Query Data Instances to Include section, complete the following steps:
   - Click in the top field and choose the data class of the instance you want to include.
Click Query.

Choose the instance you want from the pop-up list and click the OK button in the list. Information about the instance is set in the InsKey, Label, and ObjClass fields. You can edit the value of the Label field, but do not change the values in the InsKey or ObjClass fields. For example, Figure 4-8 shows the entry for an operator named sampleuser@samples.com.

![Figure 4-8. Example of a Queried Data Instance Listed in a Product Rule](image)

8. (Optional) On the Installation tab, complete the fields as described in Figure 4-9.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Me</td>
<td>Optional. The stream name of an HTML rule (of type Rule-Obj-HTML) to be displayed after the application is uploaded. This HTML rule must belong to one of the RuleSets and versions listed on the Contents tab, with Rule-Admin-Product as the Class Name key part.</td>
</tr>
<tr>
<td>Install Activity</td>
<td>Optional. An activity to be run after the application is uploaded. This activity must belong to one of the RuleSets and versions listed on the Contents tab, with Rule-Admin-Product as the Applies To class.</td>
</tr>
</tbody>
</table>

![Figure 4-9. Installation Tab Fields on the Product Form](image)

9. Complete the History tab with a full description of the product rule and information on its use.

10. Save the rule.
Create the Product Archive

You can now use this rule to create a product archive either by clicking the Create Zip File button on the Contents tab, or by using the Export Rules/Data wizard.

To create the archive from the product rule itself, complete the following steps:

1. Verify that all the rules are checked in, including this product rule. (View > Rules > Checkouts.)
2. On the Contents tab of the product rule, click Create Zip File.
3. Enter the file name and click OK. The system displays the message **Getting records from the database. Please wait...**, and then an in-progress display appears. (This step can take several minutes to finish.)
4. To confirm that the operation was successful, use the Class explorer to examine the most recent instance of Log-PegaRULESMove.

To create the archive by using the Export Rules/Data wizard, see “Creating a ZIP File from a Product or Patch Rule” on page 4-22.

Creating a Product Patch Rule

You use product patch rules to supplement a product. For example, if you have revised or new rules that need to be moved to a system but you do not want to import the whole product archive again, you can create a product patch that contains a subset of only new or changed rules.

Typically, the RuleSet section of a product rule specifies the major version or the major version and minor version only; for example: “PegaSample-IntSvcs 04” or “PegaSample-IntSvcs 04-02.” By using the RuleSet section of a product patch rule to specify all three parts of a version (the major version, minor version, and patch version, “PegaSample-IntSvcs 04-02-03,” for example), the product patch rule gathers only rules that were added in that version.
Creating a product patch rule is similar to the process of creating a product rule, with the following exceptions:

- The New form also requires a revision number as a third key part.
- You must identify the data instances with a When rule. The patch rule does not contain a Query Data Instances section.

Before you can define a product patch rule, you need to know the name and version of an existing product rule, the revision level to create, the RuleSets and versions to include, and the data instances to include. You must also create the appropriate When rules (Rule-Obj-When).

Complete the following steps:

1. From the Rules by Type explorer, choose SysAdmin > Product Patch. A list of instances appears.

2. Click New. The New form appears.

3. Enter the name, version, and patch version of the product. Choose the RuleSet name and version.


5. On the Contents tab, specify the RuleSets and versions.

6. In the Data to Include section, specify the data class of the instance you want to include, and then choose the appropriate when rule. Include a note that describes the data instances being identified by the when rule.

7. On the Installation tab, complete the fields as described in Figure 4-9.

8. Complete the History tab with a full description of the product revision and its use.

9. Click Save.
Create the Product Patch Archive

You can now use this rule to create a product patch archive, either by clicking the Create Zip File button on the Contents tab, or by using the Export Rules/Data wizard.

To create the archive from the product patch rule, complete the following steps:

1. Verify that all the rules are checked in, including this product patch rule. (View > Rules > Check Outs.)

2. On the Contents tab of the rule, click Create Zip File.

3. Enter the file name and click OK. The system displays the message **Getting records from the database. Please wait...**, and then an in-progress display appears. (This step can take several minutes to finish.)

4. To confirm that the operation was successful, use the Class Explorer to examine the most recent instance of Log-PegaRULESMove.

To create the archive by using the Export Rules/Data wizard, see “Creating a ZIP File from a Product or Patch Rule” on page 4-22.

Exporting Rules and Data with the Wizard

You can use the Export Rules/Data wizard to do the following:

- Create a ZIP file from a RuleSet.
- Create a ZIP file from an existing product or patch rule.
- Download a previously created ZIP file to a local directory on your workstation.

This section describes each option.
**Creating a ZIP File from a RuleSet**

Complete the following steps:

1. From the menu, choose File > Export Archive.

2. In the Select Export Mode field, choose By RuleSet/Version.

3. Choose the RuleSet from the RuleSet field. Choose the desired version from the RuleSet Version field.

4. Enter a file name for your ZIP file in the File Name field. The .zip extension is not required.

5. Click Create Zip File. The system displays the message **Getting records from the database. Please wait...,** then the progress of the wizard.

6. When the wizard finishes, it displays a final status message and a link to the ZIP file. Click on the link to download the file to a local directory on your workstation.

If you export a RuleSet from which rules are checked out, they are counted as errors and logged as instances of Log-PegaRULESMove-Error. If there are errors, click on Total Number of Errors in the lower right corner of the display form to see the error message(s).

**Creating a ZIP File from a Product or Patch Rule**

Complete the following steps:

1. From the menu, choose File > Export Archive.

2. In the Select Export Mode field, choose By Product or By Patch, as appropriate.

3. Choose the rule and version.
4. Enter a filename for your ZIP file in the File Name field. The .zip extension is not required.

5. Click Create Zip File. The system displays the message **Getting records from the database. Please wait...**, then the progress of the wizard.

6. When the wizard finishes, it displays a final status message and a link to the ZIP file. Click on the link to download the file to a local directory on your workstation.

**Downloading a Previously Zipped ZIP File**

As mentioned, all ZIP files created from a product rule, a patch rule, or the Export Rules/Data wizard are stored in the Process Commander ServiceExport directory. You can download any archive currently located in the ServiceExport directory with the Export Rules/Data wizard.

Complete the following steps:

1. From the menu, choose File > Export Archive.

2. In the Select Export Mode field, choose Zip File on Server.

3. Choose the file to download and click Download Zip File. The standard Windows File Download window appears. (The tool only locates files with the extension “.zip” in lowercase letters.)

4. Specify a name and location for the downloaded file and click Save. The ZIP file is saved to your system.
Importing Rules and Data

When you import rules into the PegaRULES database, you are extracting XML data from a ZIP file and inserting it into the PegaRULES database as rules and data instances.

When importing multiple RuleSets, be aware of any possible dependencies that rules in one RuleSet may have on rules in another. RuleSets should be loaded in dependency order: load fundamental RuleSets before loading dependent RuleSets. For example, Pega-RULES is the most fundamental RuleSet in Process Commander; Pega-WB depends on Pega-RULES. Therefore, the Process Commander RuleSets would be loaded in the following order:

1. Pega-RULES
2. Pega-WB
3. Pega-IntSvcs
4. Pega-ProCom

To import rules or rules and data, complete the following steps:

1. From the menu, choose File > Import Archive.

2. In the Select Import Mode field, choose Local Zip File.

3. Click the Browse button next to the File Name field to choose the file.

4. Click Upload File. The wizard uploads the file to the Process Commander ServiceDirectory directory and displays the message **File Uploaded Successfully.**

5. In the Select Import Mode field, choose Zip File on Server.

6. From the list of files, choose the one you just uploaded. The wizard examines the file and displays several options.

7. Choose one or more of the following options as appropriate for your situation:
− Compile Libraries. Choose this option when the RuleSet contains utility function or utility library rules and you want the wizard to compile the libraries as soon as the rules are loaded.

**Note**: You can recompile all libraries later by using the Extract Files option in the System Management application.

− Overwrite Existing Data. Choose this option to overwrite any data instances on the target system that have the same keys as an instance in the ZIP.

− Overwrite Existing Rules. Choose this option to overwrite any rules on the target system that have the same key (pzInsKey) as a rule in the ZIP.

− Execute Install Activity. When the ZIP is an archive generated from a product or patch rule, choose this option to run the installation activity specified by the rule as soon as the rules are moved.

8. Click Import.

If records in the ZIP file already exist on the target system, but the appropriate overwrite option was not selected, these records are skipped and are not imported. If there are errors, click on **Total Number of Errors** in the lower right corner of the display form to see the error message(s).

9. Although the new rules and data objects are present on the target system, users may not see them until their access rights have been adjusted. Adjust your access group as necessary to provide yourself access to the uploaded rules, data, work pools, and application rules.

10. Log off and then log back in with the adjusted access group to review the results.
**Rebuilding Libraries**

If the product rule, patch rule, or a RuleSet ZIP included library rules and you did not choose the Compile Libraries option when you imported the ZIP, you must rebuild the libraries now.

Complete the following steps:

1. From the menu, choose Tools > System Management Application.

2. In the System Management application, choose Administration > Rule Utility Library Extractor.

3. Enter the name of the RuleSet and the name of the library rule.

4. Click Extract Libraries. Process Commander extracts the rule, generates Java source files for the rule, and compiles the source file into a Java class.
Rule Maintenance

This section describes how to perform the following kinds of rule management tasks:

- Use the Revalidate feature to open and save selected rules or data instances of a certain type when you have introduced a change to the application that needs to be incorporated into other rules. For example, if you make a change to an HTML rule that all the harness rules use, you could use Revalidate to force all the harness rules to be regenerated.

- Use the Skim feature to consolidate multiple versions of a RuleSet into one new version.

- Use the Rename feature to change the name of a RuleSet or class rule when you determine that you used the wrong name when you created it.

Revalidating Rules

After implementing code or rule changes that can affect other rules, upgrades, or hot fixes, you can force bulk revalidation of rules or data objects of a single type. For example, you can revalidate all Operator ID instances, or all harness and flow rules in a specific RuleSet and version.

Bulk revalidation is similar to opening and saving each rule or data instance. If an instance fails validation – perhaps a validation rule has changed since the last time the rule was saved – the previous version of the rule or data instance remains unchanged.

To revalidate rules, complete the following steps:

1. From the menu, choose Tools > Revalidate and Save. The Revalidate form appears (Figure 4-10).
2. Choose the desired class from the Type field. If you choose a Rule- class, additional fields appear in which you can choose a RuleSet name and version and optionally move the rules to another version.

3. Enter a description in the Memo field.

4. (Optional) To choose a subset of rules in the class, click List. A list of rules in the class appears.
   - To choose a rule for updating, check the Update? box (selected by default).
   - To choose all the rules, click Check All.
   - To deselect all the rules, click Uncheck All.
   - To invert the selection list, click Inverse.
   - To clear the validation status, click Clear Status.
   - To view a rule, click the Open icon.

5. When you have selected the rules to revalidate, click Run. The selected rules are validated and resaved. (To stop the process, click Pause.) If a rule is valid, a checkmark appears in the Status column; if not, a red X appears.

**Note:** Errors are not displayed in the form. To see the error message, let the mouse hover over the X. You can also open the instance and save it to see detailed error information.
**Skimming to a New Version on the Same System**

After an application has matured, you may need to consolidate multiple versions of its RuleSets into one new version, retaining only the newest (best) versions of the rules that have changed over time. This is called “skimming.”

When you skim rules to a new version, you create a new major version of a RuleSet. This option collects the highest version of every rule in the RuleSet (except for those in classes that do not have versions, such as Rule-Obj-Class) and copies them to a new major version of that RuleSet on the same system.

For example, if you want to create version 02-01-01 of a RuleSet, skimming takes the highest version of every unblocked rule in version 01-xx-xx and copies it into a new rule in major version 02. A rule at version 01-02-15 (assuming that is the highest version of that rule) is copied to version 02-01-01.

To skim to a new version, follow these steps:

1. From the menu, choose File > Skim a RuleSet. The Skim form appears (Figure 4-11).

![Figure 4-11. Skim Rules Form](image)

2. Choose the desired RuleSet from the RuleSet selection box. Choose the desired version from the Major RuleSet Version selection box. Enter the major version number to which you want to update in the Major New RuleSetVersion field.

3. Click Skim. The system displays the message **Getting records from the database. Please wait...**, then an in-progress display appears.
If there are errors, click on Total Number of Errors in the lower right corner of the display form to see the error message(s).

**Note:** You cannot move records to an existing RuleSet version.

4. To skim another RuleSet, click Skim.

**Classes Without Versions**

Rules in these classes do not have versions and therefore cannot be skimmed:

- Rule-Access-Role-Obj
- Rule-Access-Deny-Obj
- Rule-Application
- Rule-Obj-Class
- Rule-RuleSet-Name
- Rule-RuleSet-Version
- Rule-Utility-Library

To move rules in these classes to another system, create a ZIP file either with the Export Rules/Data wizard or with a product or patch rule.

**Renaming a RuleSet**

The Rename RuleSet wizard renames a RuleSet and examines instances in other RuleSets of your application for dependencies. Use this feature sparingly because a RuleSet name affects the rules within the RuleSet, and may also affect data objects and rules in other RuleSets that refer to the RuleSet.

Before you begin, complete the following tasks:

- Verify that the RuleSet you plan to rename is not currently listed in your access group(s).
- Verify that none of the rules in the RuleSet is checked out. (View > Rules > Check Outs.)

- Use the Export Rules/Data wizard to create a ZIP archive of the RuleSet (backup the RuleSet).

To rename a RuleSet, complete the following steps:

1. From the menu, choose Tools > Rename a RuleSet.

2. Enter the RuleSet name from the Original RuleSet selection box.

3. Enter the new name in the New RuleSet field, and click Rename.

   The RuleSet and its instances are renamed, and instances in other RuleSets in the application are examined for dependencies. (This can take some time.) The wizard stores a description of what occurred as an instance of Log-Rename-RuleSet.

   The wizard enumerates the instances that are successfully renamed, the number of errors (if any), and any instances in other RuleSets that may contain the former RuleSet name.

**Renaming a Class Rule**

During the design phase of your application, your development team may choose class names that you want to change later on. If this happens, you can use the Rename Class feature to rename a class.

Renaming a class affects not only the rules within its RuleSet but may also affect data objects and rules in other RuleSets that refer to the class. The Rename Class utility renames a class and its subclasses and takes care of rules that apply to the renamed class. However, it **does not rename any saved instances of the class** – that is, data or work.

This feature is useful for developers who determine they need to rename classes **before** an application is put into production. If you choose to rename a class after
work has already been created for that class, the **work objects** will be **orphaned**.
Therefore, you should **never rename a class on a production system**.

To rename a class, complete the following steps:

1. From the menu, choose Tools > Rename a Class. The Rename a Class form appears.

2. Choose the class from the Original Class selection box, enter the new name in the New Class field, and click Rename. You will see the prompt **All classes that start with class will also be renamed. Do you wish to continue the renaming process?**

3. Click OK. The class and its subclasses are renamed, and instances in other classes of the application are examined for dependencies. (This can take some time.) The wizard stores a description of what occurred as an instance of Log-Rename-Class.

The Rename Class utility enumerates the instances successfully renamed and the number of errors (if any) and lists any instances in other classes that contain the previous class name.