Configuration Settings Reference Guide

PegaRULES Process Commander Version 6.2

Pega

This document is the property of:
Pegasystems Inc.
101 Main Street
Cambridge, MA 02142-1590

Phone: (617) 374-9600
Fax: (617) 374-9620

www.pega.com
Document: Config Settings Reference Guide
Software Version: 6.2
Updated: September 1, 2011

Contents

Chapter 1: Configuration Settings Structures

Overview

There are a number of preferences which can be set or changed for a Process Commander installation. Beginning in Version 5.1 and continuing through Version 6.x, these settings can be in several places:

- the prconfig.xml file
- the prbootstrap.properties file
- System Settings rules
- Dynamic System Settings instances

In versions of Process Commander prior to 6.2, the *prconfig.xml* file holds the preferences that can be set or changed for a PegaRULES installation. These preferences refer to variables which are shipped in the PegaRULES code; thus, application developers were not able to add additional settings or change the functionality of existing settings (except within the values allowed by the settings themselves).

Beginning in Version 5.1, developers can use System Settings to store system constants. System Settings are stored in the database (as either rules or data instances). Thus, they are read from the database by *all* nodes running that Process Commander installation. All nodes will have the same setting values, unlike the values in the prconfig.xml file, which apply only to the node where that file is located.

In Version 6.2 and beyond, most of the prconfig settings and some of the prbootstrap settings are now stored in the database as Dynamic System Settings (instances of Data-Admin-System-Settings). However, not all the settings may become Data-Admin-System-Settings; since these are stored in the database, they can only be accessed *after* the Process Commander application has been initialized and the database has been accessed. Therefore, the database access settings must remain in these files.

This document explains each type of setting, and then lists and describes all the valid entries for each type and how they can be used to modify the PegaRULES configuration.

For information on editing the prconfig.xml and the prbootstrap.properties files, and how to change them, please reference the document *Editing the prconfig.xml and prbootstrap.properties Files in Process Commander.*

System Settings

There are two System Setting classes:

- System Settings – instances of Rule-Admin-System-Settings
- Dynamic System Settings – instances of Data-Admin-System-Settings

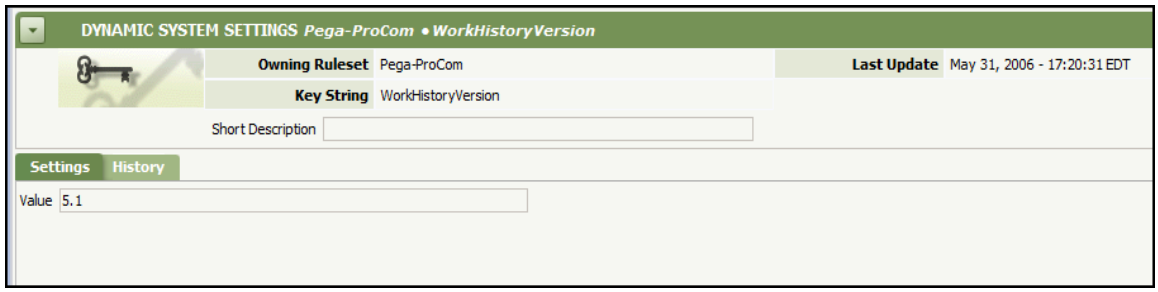
Unlike Agents, where the Agent Schedules (instances of Data-Agent-Queue) are *copied* from the Agent Queues (instances of Rule-Agent-Queue) at startup, these two System Settings classes are *completely separate*.

The classes hold different types of values: Rule-Admin-System-Settings instances hold constants; Data-Admin-System-Settings instances are created and set programmatically. If there is a reason for any program to update the setting (updated based on changeable factors when the system is moved from test to production) then a Dynamic System Setting should be used. If the system needs to reference a value (like a URL), but that value will not change as a result of processing, then a Rule-Admin-System-Settings instance should be used. The application developer should carefully consider the type of functionality they wish to implement before choosing one type or another.

Dynamic System Settings – Version 6.1 and prior

The Dynamic System Settings (instances of Data-Admin-System-Settings) are *not* rules, so they do not use Rule Resolution. As instances of the Data- class, it is not possible to have more than one Dynamic System Setting with the same name.

The **Owning RuleSet** key field is available, to provide an additional namespace qualifier. (As these are Data instances, they are not stored in a RuleSet, but can be qualified by a RuleSet Name.)



The **Key String** field holds the name of the System Setting, and should be a descriptive word or phrase with no spaces. This can be a single word, as in the example above. The forward slash ("/") is also valid in this field, and can be used to form compound names (again, with no spaces), in order to help sort settings into logical groupings. For example, there are several settings which relate to the full-search index, and are labeled as such: "indexing/enabled," "indexing/hostname," "indexing/ruleenabled," etc.

IMPORTANT: As these are data instances, many are *not shipped* with the standard Process Commander product. Some of these instances will subsequently be created programmatically by the installation process, or by other system processes.

Due to the fact that they are not shipped as a standard part of an application's configuration, the Dynamic System Settings do not have the five-level structure that is available in the System Settings rules. Instead, there is only one value.

Dynamic System Settings can be set to shipped values, or can be created and then set programmatically later through activities.

Dynamic System Settings – Version 6.2 and beyond

Prior to Version 6.1, the *Key String* field for a Data-Admin-System-Settings instance (the pyPurpose) used to include just the **setting name**.

Example: *indexing/RuleEnabled*

Beginning in Version 6.2, the default functionality has changed: the *Key String* has been expanded, and can include additional information:

- file type *(new)*
- setting name
- classification *(new)*

The *file type* is the first part of the new Key String, and specifies the kind of setting this instance is. There are several possible file types:

- **prconfig** – for the prconfig settings
- **properties** – for the prbootstrap.properties settings
- *blank* – no value at all, for nonspecified Data-Admin-System-Settings

The *setting name,* in the middle of the Key String, is the name of the setting (obviously), and can include long strings, divided by slashes or periods.

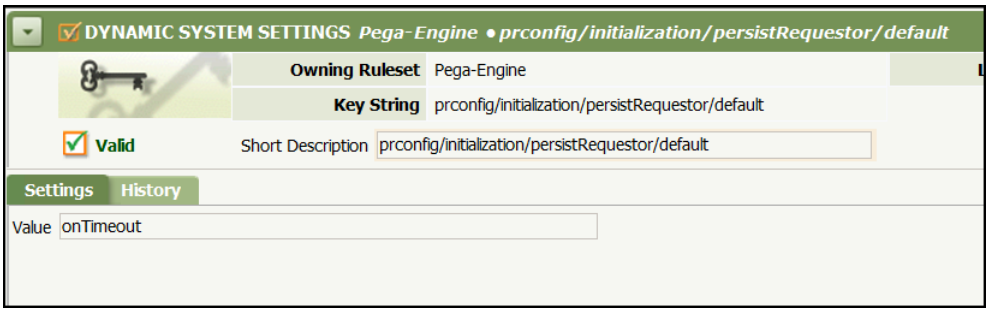
- prconfig example: *alerts/connecters/outboundMappingTime/enabled*

- prbootstrap example: *com.pega.pegarules.bootstrap.codeset.version.Pega-EngineCode*

The *classification* is the last part of the Key String; it identifies the node classification. (*Classifications* are described in *Changes in Version 6.2,* later in this document.)

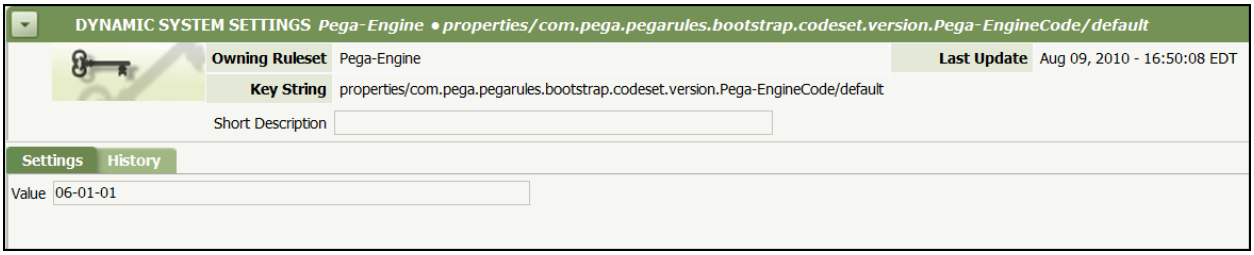
Important: All configuration settings (from prconfig and prbootstrap) will have an *Owning RuleSet* of Pega-Engine. Config settings with any other *Owning RuleSet* value will be ignored. (Otherwise, the same config setting could be defined with two different *Owning RuleSet* values, and there is no inheritance hierarchy for Data- to determine which value to use in each situation.)

Here is an example of a Data-Admin-System-Settings entry for a prconfig value:



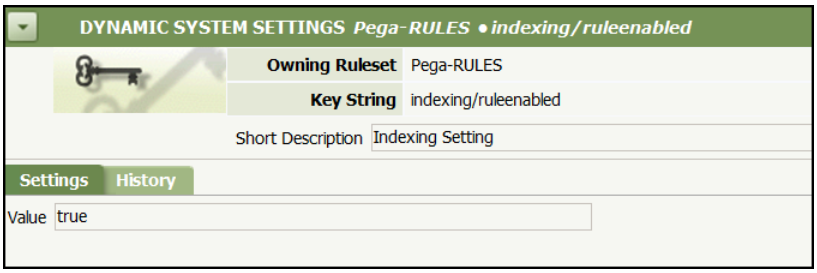
- file type = “prconfig”
- setting name = “initialization/persistRequestor”
- classification = “default”

Below is an example of a Data-Admin-System-Settings entry for a prbootstrap.properties value:



- file type = “properties”
- setting name = “com.pega.pegarules.bootstrap.codeset.version.Pega-EngineCode”
- classification = “default”

As a comparison, here is an example of an original Data-Admin-System-Settings setting. These types of settings are still valid in Version 6.2:



- file type = *blank*
- setting name = “indexing/ruleenabled”
- classification = *blank*

Also note that the Owning RuleSet is *Pega-RULES*, not Pega-Engine.

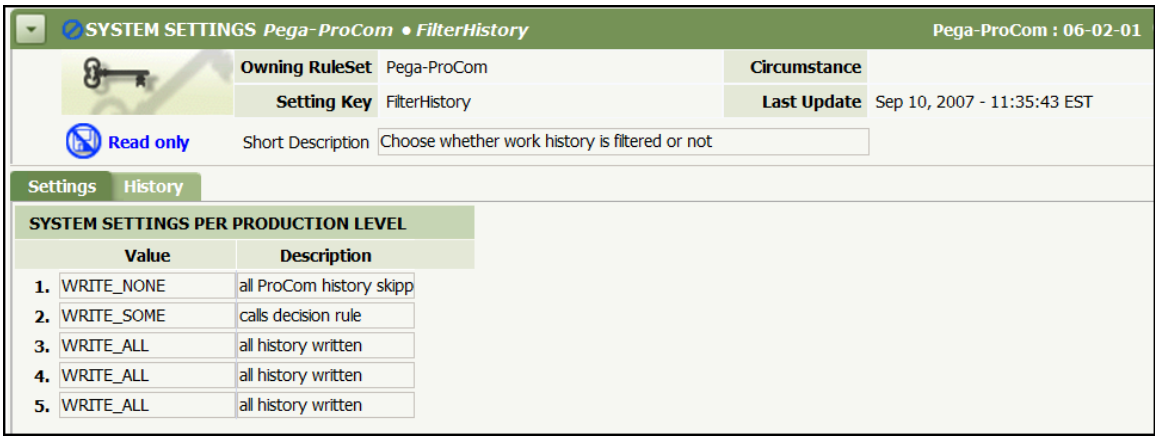
System Settings (Rule-Admin-System-Settings)

System Settings (instances of Rule-Admin-System-Settings) are **constants** stored in **rule** instances.

As with other rules, the System Settings which are shipped in the standard ProCom RuleSets are locked and cannot be changed. However, as they are rules, Rule Resolution will apply; if they need to change a value for the System Settings, application developers can save the System Setting rule to a higher-level RuleSet in the application and change its value. When the application is run, the rule in the higher-level RuleSet will override the one in the locked PegaRULES RuleSet (as always).

Application developers could also create their own System Settings in the application RuleSets. These settings should be considered as constants for that application (or that part of the application); they cannot be changed programmatically (by running an activity), but only by going into the rule and updating it.

Sample System Settings rule:



The **Setting Key** field holds the name of the System Setting, which should be a descriptive word or phrase with no spaces. This can be a single word, as in the example above. The forward slash ("/") is also valid in this field, and can be used to form compound names (again, with no spaces), in order to help sort settings into logical groupings. For example, there are several settings which relate to the full-search index, and are labelled as such: "indexing/enabled," "indexing/hostname," "indexing/ruleenabled," etc.

The **Owning RuleSet** is the RuleSet which "owns" the code that asks/calls for this System Setting instance (the RuleSet where the code is stored). Although the rule

must be saved into a RuleSet, the Owning RuleSet field is made part of the key to this setting to provide namespace qualification. In the RuleSet where a setting is originally defined, the RuleSet and the OwningRuleSet will be the same. When that setting is overridden in a customer's RuleSet, the OwningRuleSet remains unchanged, although the RuleSet where the setting is now saved is the name of the customer's RuleSet. This prevents problems when the same named setting is created in two different RuleSets.

Example:

For example, Process Commander ships a System Setting called "FilterHistory" stored in the Pega-ProCom RuleSet. When writing the application for Acme Co., the developer also created a System Setting called "FilterHistory" which had a different functionality, and saved it to the AcmeCo RuleSet. This is fine: although these two settings have the same name, they are stored in different RuleSets.

However, if for some reason the developer later had to overwrite the Pega-ProCom setting "FilterHistory" and wanted to store that in the AcmeCo RuleSet (since customers can't change items in the Pega- RuleSets), they could not do so if "FilterHistory" already existed in the AcmeCo RuleSet.

Therefore, the *Owning RuleSet* key is added. The "FilterHistory" defined in AcmeCo would have an "owning RuleSet" of AcmeCo, as that is where it was created, so it would be different from the "FilterHistory" with an Owning RuleSet of Pega-ProCom.

System Settings Per Production Level

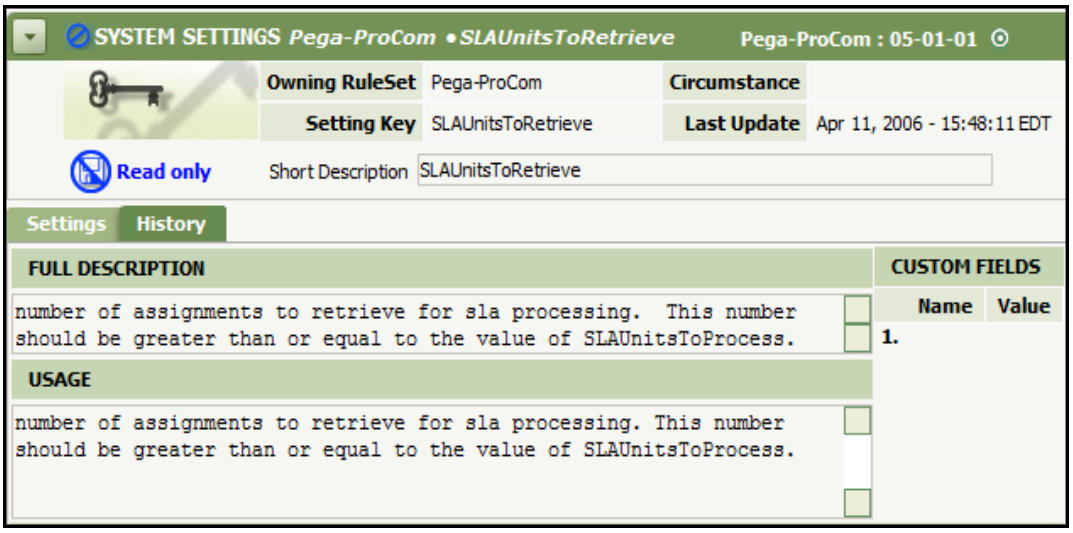
The **System Settings Per Production Level** shows the values at each production level. There are five system levels in Process Commander:

Level	Description
1	experimental
2	development
3	test
4	pre-production
5	production

Each Process Commander system has its Production Level set in a Data-Admin-System instance, depending upon its usage (development, QA, production, etc.).

The System Settings instances allow the developer to provide values for the specified System Setting for all system levels. This means that if an application developer is creating an application in a development area, and plans to move it to a QA/test area and then into production, they can provide values for this System Setting for each level, and then just move the system into each area without having to reset the rule value.

As with many rules, the **History** tab contains the **Full Description** and the **Usage** field.



For the System Settings rules, the **Usage** field should be completed with an explanation of this Setting and what it is used for, as well as any limits or other notes.

Changes in Version 6.2

settingsource setting

A new setting has been added to both of the configuration files: *settingsource*.

- prconfig.xml entry: initialization/settingsource
- prbootstrap.properties entry: initialization.settingsource

The *settingsource* entry will remain in the config files – it will not be loaded into the database - and specifies how the system should find the configuration settings.

There can be two specified values:

Setting value	Description
Merged	Default for Version 6.2 The system will check for settings in the database (Data-Admin-System-Settings) as well as the file. If settings are found in both the file and the database, ***the file setting value will take precedence.***
File	Default for Version 6.1 SP2. The system will *ignore* Data-Admin-System-Settings as entries, and use *only* the entries in the configuration file.

Classifications

For any configuration setting where the value in the file on the node is different than the value in the database, the value in the file will take precedence. If an administrator has one or two settings which they wish to change on one node only, they can do that by entering those settings into the prconfig file of that node.

However, some nodes are specifically designated to do particular kinds of processing. One or more nodes in the system might be the nodes which are configured to do agent processing, or hold the Search indexes, or run the BIX utility. These nodes would have a group of config settings which should be set to the same value for all of these type of nodes (all agent processing nodes should have agent processing

enabled, for example), but which are different from the “standard” node settings (all non-agent nodes should have agent processing disabled).

For this situation, it is now possible to create a *classification*, which identifies the type of node, to use in grouping the settings.

There are no default classifications. Some, like “agent” or “search,” will be common to many customers. Customers can also designate their own classifications, based on different applications being run, or whether that server is old and has slower hardware (and thus needs different settings to make it as efficient as possible), or whatever differentiators they wish to highlight for their nodes.

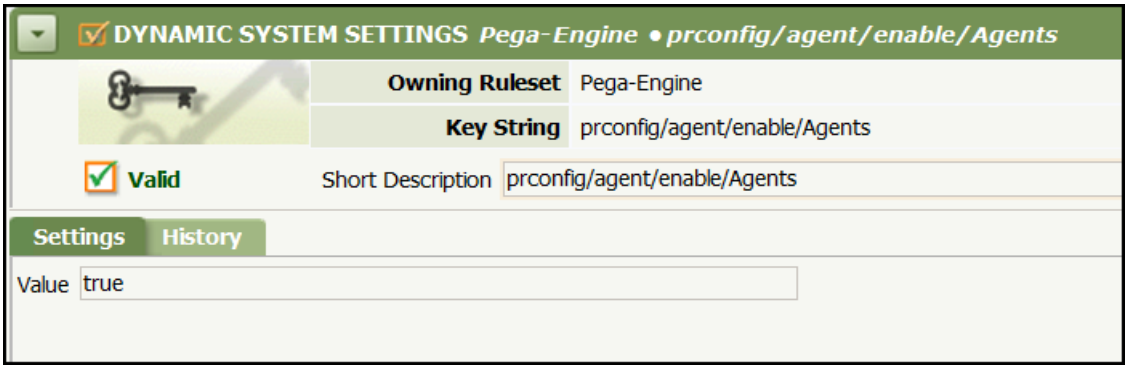
The classification is defined in the prconfig.xml file, and then referenced in Data-Admin-System-Settings entries. A new prconfig.xml setting has been defined: *nodeclassification.*

Example:

```
env name="initialization/nodeclassification" value="Agents"
```

The NodeClassification value is also part of the Data-Admin-System-Settings key, so settings uploaded to the database from one of these specially-designated nodes can be differentiated from settings in the “default” node.

Example:



The Data-Admin-System-Settings entries which have a Classification are only valid for nodes which have that same classification specified in the *nodeclassification* setting for their prconfig.xml file. So the above Data-Admin-System-Settings value would *not* be used for:

- nodes where the prconfig.xml file does not have a *nodeclassification* setting defined

OR

- nodes where the *nodeclassification* setting has a different value than “Agents” (example: “Search”)

Remember that the prconfig file is **node**-specific, so any classification would have to be node-based. Thus, a node that runs four different applications could not have application-specific classifications, because they all run on the same node.

Changes to prbootstrap.properties file

In the prbootstrap.properties file, only the *CodeSet Version* and the *CodeSet Patch* entries will become Dynamic System Settings. All other prbootstrap settings will remain in that file.

Changes to prconfig.xml file

All but a few prconfig settings from 6.1 GA and prior can become Data-Admin-System-Settings entries. Certain settings are exceptions, and must remain in the original files – specifically, the settings which the system uses to find the database at startup, or where to look for the rest of the settings (database or file). All the entries which begin with *database/databases/pegarules* must remain in the prconfig file, and cannot be uploaded to the database.

nodeclassification setting

The *nodeclassification* setting defines the classification of this node. For details, please see the *Classifications* section above.

Notes on prconfig changes:

- **The settings in the prconfig.xml file *override* any setting values in the database**, whether *settingsource* is set to *file* or *merged.* This functionality allows the administrator to make a specific change to one node that will not be duplicated across the system. For example, there may be one old server in the system (due to be replaced "real soon now") which, due to its processor being much smaller than the rest of the machines, needs some different prconfig settings. These would be set on this node only.

- **There can be duplicate settings defined in Data-Admin-System-Settings with different classifications.** There might be two Dynamic System Settings defined for *agent/enabled:* one with the *default* node classification, and one with the *Agent* classification. In this example, for the node which has the *nodeclassification* setting in prconfig set to *Agent*, the *Agent* Data-Admin-System-Settings will override the value of the default node Data-Admin-System-Settings. However, if the *nodeclassification* entry is set to some other value ("Search"), or is not set (so it is automatically set to *default*), then the *default* Data-Admin-System-Settings value would apply to this node.

  All node types will use the *default* settings, unless there is a duplicate setting specific to that node classification also present, in which case the more specific setting is used.

- **Only a few settings will be defined in Data-Admin-System-Settings.** Not all prconfig settings are included in the file shipped with PRPC (in fact, only a dozen or so from several hundred that are in the code are shipped). Similarly, in Version 6.2, only those same dozen settings will be shipped as Data-Admin-System-Settings – the several hundred others won't be shipped. As with pre-6.1 functionality, any configuration setting which is not defined will use its default value (hard-coded in the engine). If an administrator needs to add a setting, they would define it as a Data-Admin-System-Settings (just as they used to have to define it in the prconfig file).

- **Customers can add new or change existing prconfig settings through Data-Admin-System-Settings** just as they would have added them in the file previously.

Upgrading Configuration Settings from Prior Versions to Version 6.2

Version 6.2 of PRPC will ship with the "minimal" prconfig.xml and prbootstrap.properties files. However, customers who are upgrading to 6.2 from an existing prior version, must upload their config files to the database manually.

There are several steps to this process, and they are different for the prconfig and the prbootstrap files.

prbootstrap.properties file

1. Upload the file using the *runPega* script. (See next section for how to execute *runPega.*)

2. ***For each node,*** edit the prbootstrap.properties file:
    - add the *settingsource* entry with a value of *merged*
    - remove all the CodeSet and Version entries, as these were uploaded to the database

prconfig.xml file

As there are many more settings in prconfig, the process to upload it is more complex, as the settings must be carefully compared across the nodes in the system.

1. **Analyze existing prconfig files.**

    - Go through all the prconfig files of the existing nodes of your system, and compile a list of settings used in each.
    - Determine what settings should be kept, what should be added, and what should be deleted, to come to a "default" representation of the settings in this file.

NOTE: Some of these settings might be "specialized" nodes, such as the Agent node or the Search node. Data on such nodes would have a different classification than the default node, and should be evaluated and uploaded separately.

2. **Create a generic prconfig.xml file.**

Once the analysis is done, create a generic prconfig that includes all the settings desired for the "default" nodeclassification.

Also create a generic prconfig file for each of the specialized nodes, with the settings that should be used for those nodes. In this generic file, *only* include settings with values which are *different* from the generic default prconfig. If the value for the a setting on default node and the specialized node are the same, do not include it in the specialized prconfig; it will not be uploaded, as it is a duplicate.

Example 1:

default node entry: `<env name="agent/enable" value="false" />`

specialized node entry: `<env name="agent/enable" value="true" />`

Include this entry in the specialized node prconfig.

Example 2:

default node entry: `<env name="initialization/explicitTempDir" value="C:\temp" />`

specialized node entry: `<env name="initialization/explicitTempDir" value="C:\temp" />`

***Do not** include this entry in the specialized node prconfig – it has the same value as the default, and the specialized node can use the default value.*

3. Upload the generic file using the *runPega* script.
(See next section for how to execute *runPega.*)

4. After uploading the generic file, start each node to determine optimal settings.

When each node is started *without* the *settingsource* value in the prconfig, an example prconfig file will be printed to the Pega log file for that node. Take this example and determine whether targeted changes for the node are necessary to accommodate obsolete settings.

5. Change the prconfig.xml file of each node.

In the prconfig.xml file of *each* node:

1. Remove all the entries which are now Data-Admin-System-Settings entries (unless they are targeted changes).

2. Add the *settingsource* entry (set to the *merged* value), along with the targeted changes required for that node (if necessary).

Executing *runPega*

Uploading the configuration settings from the prbootstrap.properties file or the prconfig.xml file to the database is done manually, using the **runPega** script via a command-line interface.

Both the Windows .bat file and the Unix .ksh file variants of **runPega** are located on the PRPC 6.x Distribution DVD, in the *scripts* directory. runPega takes three standard arguments and the name of the class to execute; to load the configuration settings, additional arguments are required.

Standard arguments

1. Location of the JDBC driver .jar file. This argument is preceded by **--driver=** and may not have spaces in the argument or the directory path.
Example:

```
--driver=E:\apache-tomcat-6.0.26\lib\sqljdbc4.jar
```

2. Location of prweb. This is the web context root name for PegaRULES. This argument is preceded by **--prweb=** and may not have spaces in the argument or the directory path. Example:

```
--prweb=E:\apache-tomcat-6.0.26\webapps\prweb
```

3. Location of configuration file. This is the file which contains the properties which PegaRULES uses to access the database containing the engine classes and assembled classes. This argument is preceded by **--propfile=** and may not have spaces in the argument or the directory path.
Example:

```
--propfile= D:\Tomcat\prbootstrapLOAD.properties
```

NOTE: This is *not* the configuration file that will be uploaded, but is a separate file specifically configured to use with *runPega.* Please reference the *Prerequisites* section of the *Loading Custom Jar Files* document for details on how to edit this file.

Standalone Class to Execute

Different executions of *runPega* require that the appropriate tool class be specified. To load the configuration settings, use the ConfigSettingsDBWriter class:

```
com.pega.pegarules.exec.internal.basic.config.ConfigSettingsDB
Writer
```

Additional arguments:

1. **Name of file or files to upload**. This would be the fully-qualified path and filename for the prconfig file and/or the prbootstrap.properties file which will be loaded into the database. Both config files may be loaded in the same script, or they may be loaded separately. (If both files need to be loaded, it is easier to do them together, as the system must be restarted each time the *runPega* script is used.)

IMPORTANT: Be sure to point to the generic prconfig that has been analyzed, not the prconfig for only one node.

Example:

```
E:\apache-tomcat-6.0.26\webapps\prweb\WEB-
INF\classes\prbootstrap.properties

E:\apache-tomcat-6.0.26\webapps\prweb\WEB-
INF\classes\prconfig.xml
```

2. **–forced parameter**.

This parameter determines whether the system will overwrite existing Data-Admin-System-Settings. (See the *Using the forced parameter* section later in this document.)

The full command-line call would look similar to the following:

```
E:\0601\Distribution\scripts>runPega.bat
--driver=E:\ apache-tomcat-6.0.26\lib\sqljdbc4.jar
--prweb=E:\apache-tomcat-6.0.26\webapps\prweb
--propfile= D:\Tomcat\prbootstrapLOAD.properties
```

```
com.pega.pegarules.exec.internal.basic.config.ConfigSettingsDB
Writer
E:\apache-tomcat-6.0.26\webapps\prweb\WEB-
INF\classes\prbootstrap.properties  E:\apache-tomcat-
6.0.26\webapps\prweb\WEB-INF\classes\prconfig.xml
-forced
```

Notes on running this script

- After the *runPega* script is run, the system must be stopped and restarted for the new settings to take effect.

- When the script is run, the console will display the list of settings in the file, and label them:
  - Uploaded to the database: "change"
  - Left in the file (value different than existing value in database): "skip"

  Example:

```
The following updates were made to the classification 'default':

changes for 'prbootstrap.properties'
  skip   'maxActive' (leave setting in file)
  skip   'maxIdle' (leave setting in file)
  skip   'com.pega.pegarules.bootstrap.engineclasses.dbcpsource' (leave setting
in file)
  skip   'poolPreparedStatements' (leave setting in file)
  skip   'dev11g.oracle.username' (leave setting in file)
  change 'com.pega.pegarules.bootstrap.codeset.version.Pega-EngineCode' from
'06-02-01' to '06-01-01'
  skip   'oracle.jdbc.class' (leave setting in file)
  skip   'maxWait' (leave setting in file)
  skip   'dev11g.oracle.password' (leave setting in file)
  skip   'dev11g.oracle.url' (leave setting in file)
  skip   'com.pega.pegarules.bootstrap.assembledclasses.dbcpsource' (leave
setting in file)
changes for 'prconfig.xml'
  skip   'database/databases/pegarules/username' (leave setting in file)
  change 'initialization/usenativelibrary' from 'true' to 'true'
  change 'identification/systemname' from 'pega' to 'pega'
  change 'database/storageversion' from '6' to '6'
  skip   'database/databases/pegarules/password' (leave setting in file)
  change 'initialization/explicittempdir' from 'c:/temp/prpc-development/dev-
prconfig/standalone' to 'c:/temp/prpc-development/dev-prconfig/standalone'
  change 'database/drivers' from 'oracle.jdbc.OracleDriver' to
'oracle.jdbc.OracleDriver'
  change 'database/basetable/name' from 'pr4_base' to 'pr4_base'
  skip   'database/databases/pegarules/url' (leave setting in file)
Import complete. 7 settings were updated.
```

- Only one configuration file of each type may be uploaded. If multiple prconfig files (for example) are named in the *runPega* script, the following error will result:

  ```
  (.config.ConfigSettingsDBWriter) ERROR   - Only one
  prconfig.xml file can be given to this tool.  No settings were
  imported into the database.
  ```

  To upload both a generic default prconfig file and a specialized prconfig file, execute *runPega* separately for each file.

- If the *runPega* utility is run on a file which is *not* a configuration file, the following error will be written to the console log file:

  ```
  (.config.ConfigSettingsDBWriter) ERROR   -
  ..\..\..\workspace\prwebj2ee\prxml\testConfigFiles\greek1-
  3_3_8-prconfig.txt is not a valid prconfig.xml or
  prbootstrap.properties file. No settings were imported into
  the database
  ```

- If an invalid configuration file is specified for the script, the following error will be sent to the log:

  ```
  (basic.config.ConfigReader) ERROR   - Error processing
  prconfig.xml file.  File may be malformed.
  org.xml.sax.SAXParseException: Content is not allowed in
  prolog.
  ```
  (*traceback follows)*

  See Appendix A for a flow on the decision process of whether settings are uploaded to the database (to prevent duplication).

Using the –*forced* parameter

If a user has previously uploaded a configuration file to the database, there may be existing Data-Admin-System-Settings entries for particular settings. Using the –forced parameter with the runPega script causes the configuration settings currently being uploaded to overwrite the ones previously stored in the Dynamic System Settings. In this case, a message is also written to the console log, specifying which settings were overwritten, together with the prior and current values:

prconfig example:

```
(.config.ConfigSettingsDBWriter) WARN    - There were conflicts between
the settings in the database and the settings in the prconfig file.
(.config.ConfigSettingsDBWriter) WARN    - The following prconfig
settings were overridden in the database:
(.config.ConfigSettingsDBWriter) WARN    - Setting Name      Value in
File   Value in Database (Will be overridden)
(.config.ConfigSettingsDBWriter) WARN    - database/storageversion
      6     5
```

prbootstrap example:

```
(.config.ConfigSettingsDBWriter) WARN    - There were conflicts between
the settings in the database and the settings in the prbootstrap file.
(.config.ConfigSettingsDBWriter) WARN    - The following prbootstrap
settings were overridden in the database:
(.config.ConfigSettingsDBWriter) WARN    - Setting Name      Value in
File   Value in Database (Will be overridden)
(.config.ConfigSettingsDBWriter) WARN    -
com.pega.pegarules.bootstrap.codeset.patch.Pega-EngineCode  2015-05-15
18.00 EST   2012-05-15 18.00 EST
```

If the *–forced* parameter is *not* used, and the system finds an existing Data-Admin-System-Setting with the same keyname (“compiler/defaultClasses”) as a setting in the file being uploaded, then the process will stop, no settings will be uploaded to the database, and a message will be displayed on the console screen:

```
NO SETTINGS WERE IMPORTED because some settings in the provided config files
are already present in the database. If you wish to change the values currently
in the database, re-run this tool with the '-forced' flag.

If you force updates, the following changes will be made to the classification
'whatever':
```

Chapter 2: prconfig Category and Environment settings

IMPORTANT: IF THESE SETTINGS ARE ADDED AS DATA-ADMIN-SYSTEM-SETTINGS ENTRIES, THEY MUST USE THE "PRCONFIG" FILE TYPE. SEE "DYNAMIC SYSTEM SETTINGS – VERSION 6.2 AND BEYOND" SECTION ABOVE.

agent Category

The agent category contains configuration settings for the Agent Manager and master agents.

enable

Type: boolean

Default: true

Functionality: Setting this entry to *true* enables application agents to run on this category.

Setting this entry to *false* will disable all agents.

Available since: Version 4.1

minimumwakeup

Type: integer

Default: 30

Functionality: Agents wake up at regular intervals to complete the task they are configured to complete. That amount of time is specified as the Trigger Interval. The minimumwakup entry specifies the minimum amount of time, in seconds, that can

be set as the Trigger Interval for an agent (instance of Data-Agent-Queue). If someone attempts to specify in a Trigger Interval field a shorter amount of time than that specified by this entry, the value of this entry overrides it.

NOTE: This interval must be at least 5 seconds.

Available since: Version 4.1

newqueuewakeup

Type: integer

Default: 600

Functionality: This entry determines how often, in seconds, the master agent wakes up to scan for new or modified Rule-Agent-Queue definitions. Therefore, if changes are made to any Rule/Data-Agent-Queue instances, this period of time may pass before the changes take effect in the running system.

Available since: Version 4.1

requestortimeoutwakeup

Type: integer

Default: 60

Functionality: This entry holds the value for how often, in seconds, the master agents wakes up to scan for stale requestors.

Available since: Version 4.1

threadpoolsize

Type: integer

Default: 5

Functionality: This setting entry adjusts the size of the batch requestor pool, to provide the number of available Java threads on which agents and batch requestors may run. The maximum

number of threads is limited by operating system considerations.

NOTE: Setting this entry to zero will disable batch requestors (as no threads would be available).

Available since: Version 4.1

SystemQueue/dumpStats

Type: boolean

Default: false

Functionality: When set to *true,* this setting is used to continually run DBTrace for the Queue Manager functionality for the entire node, and to generate a node-wide database trace file that ends up being written to the ServiceExport directory. The file name starts with "queueOperations_" and is followed by a timestamp (example: "queueOperations_20060126T172956_473_GMT.txt")

IMPORTANT: This setting should be used only in very targeted ways, as enabling it will cause performance slowdowns and large amounts of file output. This setting aids in tracking any node-wide problems with the agent queues; it shouldn't be used often, and should definitely be turned off after the problem has been traced.

Available since: Version 5.4

alerts Category

This category contains settings which can cause entries to be written into the PegaRULES ALERT log file. Some of these settings are used for the Pegasystems Autonomic Event Services.

alerts/browser/interactionTimeThreshold

This threshold monitors the elapsed time the PegaRULES application takes to process an interaction. (For the purposes of these settings, an *interaction* is defined as a single HTTP request from the client to the server, and its reply.) The threshold monitors the amount of time from when the PegaRULES application receives an HTTP request to when it sends back a response to that request.

NOTE: This threshold *does not include* the time required for the request to go from the browser to the server, and for the reply to go from the server to the browser. In addition, time spent in the application server prior to handing the request to the PegaRULES application is also not tracked. (Tools provided by the application server vendor may be used to track non-PegaRULES-application elapsed time.)

The settings can help developers recognize when the amount of time the Process Commander server takes to respond to an HTTPrequest is too long.

NOTE: Prior to 5.1, these settings were in HTTP/interactionTimeThreshold.

enabled

Type: boolean

Default: true

Functionality: This reading shows whether the Interaction Time threshold has been enabled or disabled.

Available since: Version 5.1

excludeAssembly

Type: boolean

Default: true

Functionality: The first time rules are run, Rules Assembly occurs; this can skew results when measuring interaction time. This entry indicates that the warning alert should be suppressed if the system detects that Rules Assembly has occurred during the current interaction.

Available since: Version 5.1

warnMS

Type: integer

Default: 1000

Functionality: This entry contains the Interaction Time Threshold, in milliseconds. If an HTTP interaction takes longer than the specified time, a warning will be triggered, and an alert message will be written to the PegaRULES Alert log.

Available since: Version 5.1

alerts/clientbytesreturned

This threshold monitors the amount of dynamically generated output sent to the client as a response to a request. For example, a developer might have built a huge work object with multiple sections and lots of dropdown lists that contains several hundred KBytes of data. Multiplied by a few thousand users who each create a number of work objects, this large work object could cause an excessive amount of traffic on the network, as this data all gets sent from the server to the clients.

NOTE: This threshold does *not* measure the data from reports and lists (which are stored on disk) or static content.

enabled

Type: boolean

Default: true

Functionality: This reading shows whether the Client Bytes Returned threshold has been enabled or disabled.

Available since: Version 5.5

ignoredevusers

Type: boolean

Default: true

Functionality: Developers must work with different types of data than end users; their activities would normally generate many of these alerts (because developers use structures like the Class Explorer, which takes more than 200K to display). This setting allows developers to turn off tracking this threshold for anyone who has rule-checkout privileges (anyone who has a personal RuleSet), to avoid generating constant alert messages.

Available since: Version 5.5

threshold

Type: integer

Default: 204800

Functionality: This entry contains the Client Bytes Returned Threshold, in bytes. If dynamically generated output being sent to a client machineis larger than 200K, a warning will be triggered, and an alert message will be written to the PegaRULES Alert log.

Available since: Version 5.5

alerts/clipboard/listsizethreshold

This alert monitors the number of elements which are created as part of a Page List property. A large number of embedded pages may consume substantial memory, and may indicate a report design issue or a looping logic issue.

enabled

Type: boolean

Default: true

Functionality: This reading shows whether the List Size threshold has been enabled or disabled.

NOTE: Just setting this entry to TRUE will not change system behaviour. The *warnsize* or *errorsize* entry must be set as well, before the threshold will be signalled.

Available since: Version 5.5

errorsize

Type: integer

Default: -1

Functionality: This reading contains the threshold of the maximum number of elements which are allowed in a Page List value before generating the PEGA0035 alert and throwing an exception.

When the engine is evaluating the Page List threshold, *errorsize* will only be triggered if its value is set higher than zero. If zero or a negative number is specified (which is the default), or if this setting is not put into prconfig, then the system will not print out an error message and throw an exception, no matter how many elements are created for a Page List property.

Available since: Version 5.5

warnsize

Type: integer

Default: 10,000

Functionality: This reading contains the threshold of the number of elements which are allowed in a Page List value. If this value is exceeded, a PEGA0035 WARN alert will be generated in the ALERT log. If the *warntraceback* entry is set, a traceback will also be written to the log.

Available since: Version 5.5

warntraceback

Type: boolean

Default: false

Functionality: This reading shows whether to display stack trace information in the log file when the *warnsize* threshold is exceeded.

Available since: Version 5.5

alerts/connectors/clientResponseTime

Connectors make requests or send messages to external systems, typically as part of a workflow process. Whenever a Connect Rule sends a request, the system runs a "connect" activity.

If the connect activity is taking too long to run, the processing of that activity should be examined, to make sure it isn't calling too many other activities, whens, etc., in its steps.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the amount of time the external system needs to process a request passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.3

threshold

Type: integer

Default: 500

Functionality: This entry is the threshold of the time (in milliseconds) that the external system should need to process a request.

Available since: Version 5.3

alerts/connectors/inboundMappingTime

Whenever a Connect Rule receives the answer to a request from an external system, data must be mapped from that request to Process Commander properties on the clipboard.

Exceeding this threshold could be fine for a particular application, as the connector could just be passing a lot of data. If the amount of time spent is an issue, the developer should make sure the mapping in the Rule-Connect- instance is entered correctly. If a lot of data is coming in, they should also look at how the request is structured, and whether all that data needs to be included.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for mapping the data from the request passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.3

threshold

Type: integer

Default: 500

Functionality: This entry is the threshold of the time (in milliseconds) that the mapping operation should consume.

Available since: Version 5.3

alerts/connectors/outboundMappingTime

Whenever a Connect Rule sends a request, the request data must be mapped from Process Commander properties into the form the external system expects.

If the request data mapping is taking too long, then the developer should examine the Rule-Connect- instance and make sure the mappings are correct and that the data is organized efficiently.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for mapping the data for the request to the outside service passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.3

threshold

Type: integer

Default: 500

Functionality: This entry is the threshold of the time (in milliseconds) that the mapping operation should consume.

Available since: Version 5.3

alerts/connectors/totalRequestTime

The Total Request Time is usually measured as the amount of time required to process a request, from the time the user submits the request on the client side until the time the user gets their response back (the screen updates, a message states that the submission was made, etc.). For connectors, Total Request Time measures the time between when PegaRULES sends a request to an external application, to the time that application sends back a response to that request.

NOTES:
- This measurement will apply to all types of services except BPEL.
- This setting measures elapsed (total) time, not just CPU time.

This setting is enabled by default in the system to track connectors' interaction times. Other settings are shipped with *enabled* set to *false*, including:

- outboundMappingTime
- clientResponseTime
- inboundMappingTime

If the totalRequestTime threshold is passed, then these other settings should be enabled and the threshold-passing issue re-created, to pinpoint where in the connector process the problem is occurring.

enabled

Type: boolean

Default: true

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for the request passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.3

threshold

Type: integer

Default: 1000

Functionality: This entry is the threshold of the time (in milliseconds) that one connector request should consume.

Available since: Version 5.3

alerts/database

includeSQLinTraceBack

Type: boolean

Default: false

Functionality: When set to *true,* this entry adds the executed SQL to the trace list displayed in certain alerts.

Available since: Version 5.4

rowReadThreshold

Type: integer

Default: 25,000

Functionality: This reading contains the limit for the number of rows that may be returned for a list operation on the PegaRULES database list. If this threshold is exceeded, the PEGA0027 alert is written to the ALERT log.

Available since: Version 5.4

alerts/database/blobsizeread

This alert occurs when a database query retrieves a BLOB column (pzPVStream property value, also called the Storage Stream) that is larger (in bytes) than the threshold value. The alert provides a warning that memory performance of your system may be hurt when Process Commander expands the BLOB onto a clipboard page.

enabled

Type: boolean

Default: true

Functionality: This setting enables the PEGA0039 alert log entry to be written to the ALERT log if a database query retrieves a BLOB column larger than the *threshold* value. If this setting is disabled, then no alerts will be created.

Available since: Version 6.1

threshold

Type: integer

Default: 3

Functionality: This reading contains the limit for the size, in megabytes, for the compressed size of the BLOB being read.

Available since: Version 6.1

alerts/database/interactionByteThreshold

Customers may constrain the volume of data that can be loaded from the Storage Stream in the database in a single interaction. The Database Activity Threshold is designed to recognize:

- when queries are inefficiently designed, resulting in a return of too much data
- when a return technique has gone awry, and data is being loaded indiscriminately

This functionality provides the ability, early in the cycle, to detect if one of these situations is occurring. It also provides the ability to place a cap that cannot be exceeded on data return.

Both *errormb* and *warnmb* have the potential to publish the PEGA0004 alert, signalling a query to the database that causes a high level of interactions. The difference is that *errormb* will publish the alert, and write an error to the log files, and throw an exception to interrupt the current interaction. The *warnmb* value will simply publish the alert.

If either is set to <=0, then they will not take their respective actions.

NOTE: Prior to 5.1, these settings were in database/interactionByteThreshold.

enabled

Type: boolean

Default: true

Functionality: This reading shows whether the Database Activity threshold has been enabled or disabled.

NOTE: Just setting this entry to TRUE will not change system behaviour. The warnMB or errorMB entry must be set as well, before the threshold will be signalled.

Available since: Version 5.1

errorMB

Type: integer

Default: -1

Recommended: 500

Functionality: This reading contains the threshold, in megabytes, for the volume of data (measured in uncompressed bytes) that can be retrieved from the database in one interaction with the application server, before an error is displayed. If the volume of data reaches this threshold and an error is displayed, the activity currently executing will be terminated, and a stack trace generated (both in the log file and on the screen) to aid in diagnosing what request caused the error.

When the engine is evaluating the byte threshold, *errorMB* will only be triggered if its value is set higher than zero. If zero or a negative number is specified, or if this setting is not put into prconfig, then the system will not print out an error message and throw an exception, no matter what the number of bytes read is.

Available since: Version 5.1

warnMB

Type: integer

Default: 50

Functionality: This reading contains the threshold, in megabytes, for the volume of data (measured in uncompressed bytes) that can be retrieved from the database in one interaction with the application server, before a warning is displayed. If the volume of data reaches this threshold and a warning is displayed, a stack trace will be written to the ALERT log file.

When the engine is evaluating the byte threshold, *warnMB* will only be triggered if its value is set higher than zero. If zero or a negative number is specified, then the system will not print out an error message and throw an exception, no matter what the number of bytes read is.

If this setting is not put into prconfig, then the limit is 50MB; if that limit is exceeded, then the PEGA004 alert will be published.

Available since: Version 5.1

warntraceback

Type: boolean

Default: false

Functionality: This reading shows whether to display stack trace information in the log file when the *warnMB* threshold is exceeded.

Available since: Version 5.1

alerts/database/operationTimeThreshold

Type: integer

Default: 500

Functionality: This threshold measures the elapsed time (in milliseconds) spent by a requestor for a single database operation, and can help developers see when a database query is too complex or is requesting too much data.

This threshold is itself the entry; it also has a sub-entry.

NOTE: Prior to 5.1, these settings were in database/operationTimeThreshold.

Available since: Version 5.1

suppressInserts

Type: boolean

Default: true

Functionality: This setting enables or disables the display of all SQL insert data in every entry in the PegaRULES ALERT log for database queries. By default, all entries in the ALERT log show all data associated with the alert, including customer ID numbers, passwords, and other sensitive data. Setting this entry to true will prevent a customer's sensitive data from being written to the ALERT log, which is a clear text file.

Available since: Version 5.1

alerts/declarative/lookupTime

This alert indicates that the elapsed time to load a declarative network for a given class exceeds a threshold.

enabled

Type: boolean

Default: true

Functionality: This setting enables the PEGA0024 alert log entry to be written to the ALERT log if the elapsed time to load a declarative network for a given class has exceeded the *threshold* setting. If this setting is disabled, then no alerts will be created.

Available since: Version 5.3

thresholdMS

Type: integer

Default: 1000

Functionality: This reading contains the threshold for the amount of time, in milliseconds, for the time to load the declarative network.

Available since: Version 5.3

alerts/fua/assemblytime

threshold

Type: integer

Default: 400

Functionality: This reading contains the threshold for the amount of time, in milliseconds, that it took to assemble a rule. If this threshold is exceeded, the PEGA0037 alert is written to the ALERT log file.

The following actions in the system will be counted toward this threshold:

- elapsed Java assembly time
- compile process time

Available since: Version 6.1

alerts/fua/invalidation

threshold

Type: integer

Default: 10

Functionality: This reading contains the threshold for the number of rule assembly cache invalidations that may occur before the PEGA0032 alert is written to the ALERT log file. The following actions in the system will be counted toward this threshold:

- Rules being removed from the global rules assembly cache
- Checked-out rules being removed from the personal rules cache
- Generated Java classes being invalidated

Available since: Version 5.5

alerts/fua/synchtime

This alert indicates that a requestor's wait time to access the rule assembly cache has exceeded a threshold in milliseconds.

enabled

Type: boolean

Default: false

Functionality: This setting enables the PEGA0038 alert log entry to be written to the ALERT log if a requestor's wait time to access the rule assembly cache has exceeded the *threshold* setting. If this setting is disabled, then no alerts will be created.

Available since: Version 6.1

threshold

Type: integer

Default: 400

Functionality: This reading contains the threshold for the amount of time, in milliseconds, for the requestor's wait time to access the rule assembly cache.

Available since: Version 6.1

alerts/general

This category contains settings related to alerts in general.

includeparameterpage

Type: boolean

Default: true

Functionality: This setting determines whether the parameter page of the topmost stackframe will be included in the ALERT log when the alert is generated. Depending upon what is being processed when the alert is generated, data from a work item or other sensitive records might be included in the log.

Setting this entry to *false* will prevent this sensitive data from being written to the ALERT log, which is a clear text file.

Available since: Version 5.4

stackdepth

Type: integer

Default: 5

Functionality: This setting determines the number of levels of stack frame information shown in an alert. Developers may wish to set this entry higher when troubleshooting, to get more depth of information; however, it should be reset to the default when the troubleshooting is complete - the more levels of data that are included, the faster the alert log will grow.

Available since: Version 5.4

alerts/longrunningrequests

Occasionally requestors will get "stuck" in the server. Beginning in Version 5.2, the master agent can clean these “stuck” requestors out of the system. Each time the master agent wakes up (default: 60 seconds), it checks each requestor and tries to get a lock on that requestor (sees if it's busy). If the lock attempt fails (the requestor is busy), the master agent keeps track of this requestor through several more agent intervals. If it is still busy after the *requesttime* interval, then the system will generate an alert.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the requestor is busy for the amount of time specified in the *requesttime* setting. If this setting is disabled, then no alerts will be created.

Available since: Version 5.2

requesttime

Type: integer

Default: 600

Functionality: This setting contains the amount of time, in seconds, which the system will track a busy requestor before generating an alert.

Available since: Version 5.2

notifications

Type: integer

Default: 3

Functionality: This setting contains the number of times an alert should be generated for a particular requestor. If the system determines that the requestor has been busy for the requesttime interval, it generates an alert. If another requesttime interval passes, then another alert is generated, up to the limit of this setting.

Available since: Version 5.2

alerts/memory

After a Garbage Collection has been completed, the JVM can tell exactly how much memory is being used by the system at that time. If the percent of memory used is over the set threshold, out of memory errors may occur shortly.

The Collection Usage Threshold Notification is a JMX notification from the JVM; it is available *only in Java 1.5 or later*. For systems using those versions of Java, the developer can set this threshold in the JVM. If it is not set, Process Commander will automatically set the threshold to 90% upon startup. The below entries relate to this functionality.

NOTE: The default values for these settings are correct for almost all systems. *Do not change these settings unless specifically instructed to do so by your Pegasystems representative.*

enabled

Type: boolean

Default: true

Functionality: This setting enables log entries to the ALERT log if the percent of memory used exceeds the threshold which was set (either by the customer or by Process Commander).

Available since: Version 5.4

classstoragepool/percentused

Type: integer

Default: 90

Functionality: This entry holds the threshold for the percentage of memory used by the class storage area. If the memory used by the class storage area exceeds this value after a garbage collection, a PegaRULES alert will be generated.

Available since: Version 5.4

longlivedobjectpool/percentused

Type: integer

Default: 90

Functionality: This entry holds the threshold for the percentage of memory used by the long-lived object area. If the memory used by the long-lived object area exceeds this value after a garbage collection, a PegaRULES alert will be generated.

Available since: Version 5.4

minimuminterval

Type: integer

Default: 60

Functionality: After a garbage collection has been completed, if the percent of memory used is over the set threshold, the Pega alerts will be generated. If the system is being heavily used, every interaction could generate another instance of this alert, filling the ALERT log with notifications, which may not all be necessary. Therefore, the system has been configured to write fewer notifications to the log.

This entry holds the minimum amount of time, in seconds, between the alert notifications written to the log.

NOTE: This setting is only valid for Sun or JRockit JVMs, version 1.5 or higher.

Available since: Version 5.4

overrideexistingthreshold

Type: boolean

Default: false

Functionality: If some mechanism other than Process Commander has already set a Collection Usage Threshold (the application server, the developer) for one of the memory pools, then setting this flag to *true* will override the set threshold with the value from the PegaRULES configuration file.

Available since: Version 5.4

alerts/memory/onalert

heapdumpfiledirectory

Type: string

Functionality: This entry specifies the location of the directory to which the heap dump will be written (if set in *performheapdump).*

NOTE: This setting is only used for Sun 1.6.0 (and higher) JVMs. (IBM creates its own directory.)

Available since: Version 5.4

performheapdump

Type: boolean

Default: false

Functionality: If this entry is set to *true,* then when the PegaRULES alert for Java memory usage is generated, a Java heap dump will be created.

NOTES:

- A heap dump will only be triggered the first time the alert is generated.
- The heap dump will only occur if platform support is available: IBM 1.5.0 JVMs (and higher) or Sun 1.6.0 JVMs (and higher).

- If the system is running on a Sun JVM, this heap dump will be written to the directory specified in the *heapdumpfiledirectory* entry. If the system is running on an IBM JVM, then the heap dump file will be written to the application server's current working directory.

Available since: Version 5.4

performthreaddump

Type: boolean

Default: false

Functionality: If this entry is set to *true,* then when the PegaRULES alert for Java memory usage is generated, a thread dump (stack trace from every thread in the JVM) will be written to the standard PEGA log file.

Available since: Version 5.4

alerts/parse/totalTime

When mapping data in Process Commander, the following Rule-Parse classes may be used:

- Rule-Parse-Delimited
- Rule-Parse-Structured
- Rule-Parse-XML

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for parsing data (using any of these classes) passes the threshold. If this setting is disabled, then no alerts will be created.

Available since: Version 5.1

threshold

Type: integer

Default: 500

Functionality: This entry is the threshold of the time (in milliseconds) that the data parsing operation should consume.

Available since: Version 5.1

alerts/PRThread/count

PRThreads are created each time a developer runs a wizard, or creates a new context to test-run an application (without logging out of the Developer Desktop). These PRThreads contain a clipboard and are resource intensive. The settings help developers track how many threads are created, so developers won't try to create too many threads with each process.

NOTE: The alert will be written to the log file every time the threshold is passed. If the developer created 4 PRThreads, and then creates 3 more (total of 7), the alert will be written to the log. If the developer keeps creating PRThreads, only that one alert will be present. However, if a number of the PRThreads are closed so the remaining PRThreads are below the threshold (down to 4), and then more are created (another 3, going back up to 7), then another alert will be logged.

PRThreads should be created sparingly in an application. The developer should be certain that a new thread is required; if the application is generating numerous new threads, there will be a performance cost.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the number of PRThreads created passes the threshold. If this setting is disabled, then no alerts will be created.

Available since: Version 5.1

threshold

Type: integer

Default: 5

Functionality: This entry holds the number of PRThreads that can be created before an alert is written to the ALERT log.

Available since: Version 5.1

alerts/services/activityTime

Whenever a Service Rule receives and processes a request, after the data is mapped for the response, the system runs a "service" activity.

If the service activity is taking too long to run, the processing of that activity should be examined, to make sure it isn't calling too many other activities, whens, etc., in its steps.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for running that activity passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.1

threshold

Type: integer

Default: 500

Functionality: This entry is the threshold of the time (in milliseconds) that executing the service activity should consume.

Available since: Version 5.1

alerts/services/inboundMappingTime

Whenever a Service Rule receives a request, data must be mapped from that request to Process Commander properties.

Exceeding this threshold could be fine for a particular application, as the services request could just be passing a lot of data. If the amount of time spent is an issue, the developer should make sure the mapping in the Rule-Services instance is entered correctly. If a lot of data is coming in, they should also look at how the request is structured, and whether all that data needs to be included.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for mapping the data from the request passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.1

threshold

Type: integer

Default: 500

Functionality: This entry is the threshold of the time (in milliseconds) that the mapping operation should consume.

Available since: Version 5.1

alerts/services/outboundMappingTime

Whenever a Service Rule receives and processes a request, the response data must be mapped from Process Commander properties back to the form the external system expects.

If the response data mapping is taking too long, then the developer should examine the Rule-Service- instance and make sure the mappings are correct and that the data is organized efficiently.

enabled

Type: boolean

Default: false

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for mapping the data from the response to the outside service passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.1

threshold

Type: integer

Default: 500

Functionality: This entry is the threshold of the time (in milliseconds) that the mapping operation should consume.

Available since: Version 5.1

alerts/services/totalRequestTime

The Total Request Time is usually measured as the amount of time required to process a request, from the time the user submits the request on the client side until the time the user gets their response back (the screen updates, a message states that the submission was made, etc.). For services, Total Request Time measures the time between when PegaRULES receives a service request from outside the application, until the time the application sends back a response to that request.

NOTES:

- This measurement will apply to all types of services except BPEL.
- This setting measures elapsed (total) time, not just CPU time.

This setting is enabled by default in the system to track services interaction times. Other settings are shipped with *enabled* set to *false*, including:

- inboundMappingTime
- activityTime
- outboundMappingTime

If the totalRequestTime threshold is passed, then these other settings should be enabled and the threshold-passing issue re-created, to pinpoint where in the services process the problem is occurring.

enabled

Type: boolean

Default: true

Functionality: This setting enables log entries to the ALERT log if the total amount of time required for the request passes the threshold. If this setting is disabled, then no alerts will be created.

NOTE: This setting measures elapsed (total) time, not just CPU time.

Available since: Version 5.1

threshold

Type: integer

Default: 1000

Functionality: This entry is the threshold of the time (in milliseconds) that one services request should consume.

Available since: Version 5.1

Authentication Category

The authentication category contains authentication and authorization settings.

RedirectGuests

Type: boolean

Default: True

Functionality: Process Commander sends information between the server and the client (browser) using URLs. These URLs can contain "query strings" which display calls to activities in the system or output streams of data.

Example:

```
/pr3web/PRServlet/PC-
KVSeY1NqMu72E9CQuRw%5B%5B*/!Developer?pyActivity
=ShowStream&pyBasePage=pyPortal&pyTargetStream=
FramesetDeveloper HTTP/1.1
```

PegaRULES typically redirectes guests to the standard URL format above in preparation for interactive authentication. Not all authentication schemes require this; if your authentication scheme doesn't require interaction, set this entry to *false* to reduce network traffic.

NOTE: If the initialization setting **NodeType** is set to "web", the default of the RedirectGuests setting changes to *false.*

Available since: Version 5.4

UsePreAuthenticationCookie

Type: boolean

Default: False

Functionality: By default, Process Commander generates a "cookie" for each user, to track the user's requestor ID throughout their session. When the user has logged into the system and is authenticated, the cookie will include the information that this requestor is authenticated.

This setting adds additional security to the cookie.

- If this entry is set to *false*, then the cookie will contain the same value whether the user is authenticated or not.

- If this entry is set to *true*, then PRPC will use a different cookie value when the requestor is not authenticated vs. when it is authenticated: the original cookie will be invalidated after the user is authenticated, and a second cookie with a new requestor ID will be generated and used thereafter.

This eliminates the possibility that the original cookie value could be used by an unauthenticated user to access the system.

Available since: Version 5.3 SP1

LDAP subcategory

The LDAP category is a subcategory of the Authentication category. It contains parameters that implement the authentication-only LDAP option, also known as pre-Version 4 LDAP. For information about this authentication option, see *Authentication and Integration,* posted on the Pega Developer Network.

authenticationType

Type: string

Default: *blank*

Valid Values: "simple"

Functionality: This property holds the value of the JNDI property java.naming.security.authentication.

NOTE: This entry is *not* the AuthenticationType variable that the PRAuthentication interface uses to determine whether to use internal vs. external authentication.

Available since: Version 4.1

directoryContext

Type: string

Default: *blank*

Example Values: "ou_People, dc_pegasystems, dc_com"

Functionality: The values of userIdAttribute, directoryContext, and the user ID, which are typed in by the user, make up the value of the JNDI property java.naming.security.principal

Available since: Version 4.1

initialFactory

Type: string

Default: *blank*

Example Values: "com.sun.jndi.ldap.LdapctxFactory"

Functionality: This property shows the Java class name that is used to create the object that will talk to the server designated by providerUrl. This entry holds the value of the JNDI property java.naming.factory.initial.

Available since: Version 4.1

protocol

Type: string

Default: *blank*

Example Values: *provider dependent*

Functionality: Provides the value of the JNDI property java.naming.security.protocol

Available since: Version 4.1

providerURL

Type: string

Default: *blank*

Example Values: "ldap://localhost:389"

Functionality: This entry holds the value of the JNDI property java.naming.provider.url, and shows which server is being used.

Available since: Version 4.1

userIdAttribute

Type: string

Default: "mail"

Example Values: "uid"

Functionality: This entry holds the attribute which supplies the userID for the LDAP server o look up.

Available since: Version 4.1

userNameAttribute

Type: string

Default: "cn"

Functionality: This entry contains the attribute retrieved from the LDAP server. This entry's context is the full user name which corresponds to the userID.

Available since: Version 4.1

cache Category

This category contains settings related to the retention of rule definitions in memory.

instancecountlimit

Type: integer

Default: 2000 (prior to Version 6.1)
3000 (starting in Version 6.1)

Functionality: This entry shows the initial allocation size of the Instance Cache (where the data is stored). It is also used to compute the various limits that control the maximum number of entries in the cache.

Available since: Version 4.1

classloader Category

The classloader category enables you to add Java classes to the Process Commander runtime classpath. The Process Commander classloader loads all the classes (in all the jars) it finds in the following places:

- PRGenClasses subdirectory of the temp directory – the Java generated by Process Commander.
- Process Commander ext\lib and ext\classes directories.
- Any .jar files or Java classes specified in this category of the prconfig.xml file.

The application server classloader loads the Process Commander application itself – that is, all the classes (in all the .jars) it finds in the Process Commander WEB-INF\lib and WEB-INF\classes directory (for applications installed using a WAR file) or APP-INF\lib and APP-INF\classes directory (for applications installed using an EAR file).

classpath

Type: string

Functionality: This entry specifies the location of additional Java classes or .jar files to be loaded by the Process Commander classloader. For unarchived Java classes, specify the directory that contains the classes. For .jar files, specify the full path to and name of the .jar. Use semicolons (;) to separate directory names and .jar file names.

Available since: Version 4.1

useOptimizedFileLookup

Type: boolean

Default: True

Functionality: When this entry is set to TRUE, the only class files that the PegaRULES classloader will look for are in the "com.pegarules.generated" package. All other classes referenced by the PegaRULES generated code *must* be in a .jar file.

If a customer wishes to use third-party *classes*, not in a .jar file, then this entry must be set to FALSE, which disables this important optimization. Setting this to false will cause tens of thousands of

additional File and StringBuffer objects to be generated as trash, and significant additional disk activity to search for files will occur. This is a very expensive setting; PegaRULES Best Practice strongly recommends using .jar files.

Available since: Version 4.2

collections Category

The collections category contains settings for tuning the basic caching structures.

mru subcategory

This category holds the most-recently-used levels for various caches. This level is used to rank the items in each cache according to when they were "most recently used". Items in frequent use stay near the "top" of the cache; items which are not used a lot sink to the "bottom" of the cache, where they are eventually discarded when the cache is pruned. The *instancecountlimit* levels show how many items can be kept in the cache before pruning is considered.

LowerCase / instancecountlimit

Type: integer

Default: 10000

Functionality: This entry holds the initial allocation size of the Lower Case Cache, and is also used to compute the various limits that control the maximum number of entries in the cache. The optimal size for each cache will vary based on the customer's application and rules use, and should be tuned by the application developer for maximum performance efficiency.

Available since: Version 5.1

PropertyReference / instancecountlimit

Type: integer

Default: 15000

Functionality: This entry holds the initial allocation size of the Property Reference Cache, and is also used to compute the various limits that control the maximum number of entries in the cache. The optimal size for each cache will vary based on the customer's application and rules use, and should be tuned by the application developer for maximum performance efficiency. Please reference the Caching

article on the Pega Developer Network for full details on this setting.

Available since: Version 5.1

UpperCase/instancecountlimit

Type: integer

Default: 15000

Functionality: This entry holds the initial allocation size of the Upper Case Cache, and is also used to compute the various limits that control the maximum number of entries in the cache. The optimal size for each cache will vary based on the customer's application and rules use, and should be tuned by the application developer for maximum performance efficiency.

Available since: Version 5.1

Compatibility Category

This category contains settings which allow applications written in previous versions of PegaRULES Process Commander to work with more current releases.

NOTE: Although these settings are provided, it is *strongly* recommended that developers make adjustments to their application rules in order to take advantage of current functionality, and not have to use these settings.

createMissingPages

Type: boolean

Default: True

Functionality: In Version 4.2, editing was tightened so that if a named page is referenced in the right-hand side of an expression, the page *must* exist on the clipboard. This setting will create missing named pages if they do not exist in the clipboard. The default behaviour is to set this option to TRUE, so PegaRULES will automatically create the missing pages. If set to false, the system will throw an exception if the page is missing.

Available since: Version 4.2

list42sp1

Type: boolean

Default: False

Functionality: This entry controls how the system handles blank values in an Obj-List method. The Obj-List method will do a select on data in the database, based on the information specified in the activity fields. In Version 4.2 Service Pack 1 of PegaRULES Process Commander, if the **SelectFrom** and **SelectTo** fields were blank, then the system would simply ignore that particular SelectionProperty. Thus, if there were only one SelectionProperty, and it had blank fields, the system would ignore it and return all the instances of the specified ObjClass. If there were more than one SelectionProperty, the system would ignore the blank one, and select only using the properties which had values.

Beginning in Version 4.2 Service Pack 4, if these fields are left blank, then the system will select instances of the ObjClass where the specified property has a "blank" value.

When the entry is set to *false*, the system will follow the Service Pack 4 behavior, where it selects values of the property equal to "blank." When the entry is set to *true*, then the system follows the Service Pack 1 behavior, where it will ignore any SelectionProperty where the values are blank.

For more details on this setting, please reference KnowledgeBase article #20392 – *How To Set Obj-List to Handle Blank Fields*.

Available since: Version 4.2 SP6

trimPropertyValues

Type: string

Default: "N"

Valid Values: "N" | "W" | "I" | "B"

Functionality: When records are retrieved from or written to the database, leading or trailing space characters (tabs, new-lines, and the space character) are trimmed for data going into exposed columns. However, for data going into the blob, nothing is trimmed; thus, the data will appear differently in reports, depending upon whether it is retrieved from exposed columns or the blob. This setting controls the trimming of the spaces, so that data read from or written to exposed columns can be made to match the data in the blob columns.

N – (None) means that no spaces will be trimmed, either reading from the database onto the clipboard, or saving data to the database. This entry ensures that the data in the exposed columns is the same as it is in the blob.

W – (Whitespace) is the backward-compatibility setting. All space characters will be trimmed from the front and back of data being committed to exposed columns. However, data being *retrieved* from the database to the clipboard will *not* be trimmed.

I – Data being committed to the database from the clipboard will *not* be trimmed; data being *retrieved* from the database *will* be trimmed.

B – Both data being retrieved from and committed to the database will be trimmed.

Available since: Version 5.1

warnIfLockUnlockableInstance

Type: boolean

Default: True

Functionality: Locking is enabled or disabled on each class by checking the "Allow Locking?" box on the class form. If a class has locking disabled, then trying to lock this class in an activity when this setting is TRUE will result in a WARN message being shown (in the StepStatus in the tracer). If this entry is set to FALSE, then the processing status is not set to WARN.

Available since: Version 5.1

compiler Category

The compiler category enables you to add Java classes to the Process Commander compile time classpath. The Process Commander compiler, also known as PRCompiler, loads all the classes (in all the jars) it finds in the following locations:

- The Process Commander WEB-INF\lib and WEB-INF\classes directory or APP-INF\lib and APP-INF\classes directory.
- PRGenClasses subdirectory of the temp directory – the Java generated by Process Commander.
- Process Commander ext\lib and ext\classes directories.
- Any .jar files or Java classes specified in this category of the prconfig.xml file.

compileOptions

Type: string

Default: *blank*

Example: -verbose | -progress | -inlineJSR (for javac)

Functionality: This entry specifies the command-line options to pass to compiler; the strings will vary depending upon which compiler is chosen.

The "inlineJSR" setting should be used when the customer is using the IBM JDK5. If this setting is not used, then opening a new harness (such as a work item) for the first time will cause the system to "hang" for 10 - 15 minutes while certain parts of the code compile. During this time, the JVM will consume 100% of the CPU of the system, preventing any other work from being done by any other users.

This is an IBM bug, not a PegaRULES bug. While IBM works on this issue, the impact on PegaRULES can be minimized by using this setting.

NOTE: This workaround was added into the engine code in Version 5.4. Therefore, beginning in that version, setting this entry in the prconfig.xml file is no longer necessary.

Available since: Version 4.1

compilerPath

Type: string

Default: *blank*

Example: f:/jdk1.4.2/bin/javac.exe

Functionality: This entry holds the full path and file name of the compiler executable program; the strings will vary depending upon which compiler is chosen.

Available since: Version 4.1

defaultPaths

Type: string

Default: *blank*

Example: D:/PRSearch/;D:/programs/jakarta-tomcat-5.5.16/common/lib/mq.jar

Functionality: This entry specifies the location of additional Java classes or .jar files to be loaded by the Process Commander compiler (PRCompiler). For unarchived Java classes, specify the directory that contains the classes. For .jar files, specify the full path to and name of the .jar. Use semicolons (;) to separate directory names and .jar file names.

NOTE: Beginning in Version 5.1, there is also a Dynamic System Setting called *compiler/defaultPaths*. When the classes are loaded, the .jar files listed in the prconfig.xml file setting will be loaded first, and then the .jar files specified in the Dynamic System Setting.

Available since: Version 4.1

externalClassesDir

Type: string

Default: *blank*

Example: D:/PRSearch/;D:/programs/jakarta-tomcat-5.5.16/common/lib/

Functionality: This entry specifies the path to a directory which will be added to the compile-time classpath for rules. Any .class files in this directory will be available to the compiler.

Available since: Version 4.1

externalJarDir

Type: string

Default: *blank*

Example: D:/PRSearch/;D:/programs/jakarta-tomcat-5.5.16/common/lib

Functionality: This entry specifies the path to a directory which will be scanned - all jar files found in this directory will be added to the compile-time classpath for rules.

Available since: Version 4.1

formatDisabled

Type: boolean

Default: True

Functionality: This entry enables or disables formatting of generated java source code at run-time (true = do not format). The design-time "Show Java" option will format code that compiles successfully.

Available since: Version 4.1

name

Type: string

Default: “eclipse”

Valid Values: “eclipse” | “javac”

Functionality: This entry holds the name of the compiler to use to compile Activities, etc.

Available since: Version 4.1

nameForSyntax

Type: string

Default: “eclipse”

Valid Values: “eclipse” | “javac”

Functionality: This entry holds the name of the compiler to use to check Activities compilation at save/verify time for correct syntax. The developer may specify "eclipse" or the SAME value as used with "name," but may not otherwise mix compilers (e.g. name = Eclipse, nameForSyntax=javac is invalid).

Available since: Version 4.1

systemJars

Type: string

Functionality: This entry holds a comma-separated list of jar files (located in java.home/lib) which will be used as the required system libraries. Example: "core.jar, rt.jar"

Available since: Version 4.1

verboseDiag

Type: boolean

Default: False

Functionality: For the javac compiler, this entry will enable or disable verbose output when launching a sub-process to perform the compile.

Available since: Version 4.1

crypto Category

This category contains settings which govern PegaRULES use of Java cryptography extensions (JCE) in conjunction with password obfuscation.

keyringCipherClass

Type: string

Default: "com.pega.pegarules.crypto.PRCipherV5Keyring"

Valid Values: "com.pega.pegarules.crypto.PRCipherV4Keyring"
"com.pega.pegarules.crypto.PRCipherV5Keyring"

Functionality: This entry holds the Java classname used to encrypt the keyring file. This file, in turn, is used to store database passwords in conjunction with database URLs and the prconfig.xml or pegarules.xml file.

When using an existing keyring, the system will detect whether it was created with Version 4 or Version 5, and set the default appropriately. If a new keyring is created and the keyring class is not specified, the V5 implementation will be used.

NOTE: The V4 implementation is sensitive to the directory where PegaRULES is installed and the IP address upon which the PegaRULES server is running; the V5 implementation is independent of these factors.

Available since: Version 5.1

siteCipherClass

Type: string

Example: com.pega.pegarules.crypto.PRCipherSampleBF - 128-bit BlowFish cipher

Functionality: There are two types of password encryption: a portable obfuscation algorithm, where the password value may be decrypted by any PegaRULES system installed anywhere by any customer (using the same encryption algorithm for everyone), or a reversible password obfuscated in a way that is unique to one installation. The site cipher class is the name of the Java class that

is used for a specific and unique encryption algorithm. Only systems which have this encryption algorithm would be able to decrypt passwords from that installation.

This setting specifies that Java class name, and is used if the passwords from a unique encrypted system needed to be passed to another system. There is no default for this setting – if this feature will be used, the site cipher class must be created and put into a .jar file where it is accessible to PegaRULES, and then specify the classname in the prconfig file.

Available since: Version 5.1

v5oneway

Type: boolean

Default: True

Functionality: Customers may be concerned that if a password is encrypted in the same way each time, it is subject to a dictionary attack (which checks well-known passwords). If this entry is set to TRUE, then the MD5-encoded password is further obfuscated in such a way that the encrypted string is different each time the cleartext value is encrypted.

NOTES:

- The default for this setting was changed to TRUE in Version 5.5.
- If this value is set to TRUE, then the encrypted values stored in the Data-Admin-Operator-ID record (and any other place that uses the encrypted password) will not be readable by the version 4 PegaRULES engine. Therefore, this value may only be TRUE after all of the systems sharing the PegaRULES database have been upgraded to V5; this entry will not work in a mixed-version setup (compatibility mode). If both a 4.2 and a 5.1 system will use the same database, this entry MUST be set to FALSE.

Available since: Version 5.1

v5onewaySHA1

Type: boolean

Default: false

Functionality: This setting is available for organizations whose security standards mandate the use of SHA-1 signatures. When this entry is set to TRUE, the underlying algorithm used to encrypt data for properties encrypted using v5oneway is changed from MD5 to SHA1. Examples of such properties include the Data-Admin-Operator-ID password, the Rule-RuleSet-Version password, and the Rule-Obj-Property of type *password*.

NOTES:

- This setting requires that the *v5oneway* entry (described above) be set to TRUE.
- Once the v5onewaySHA1 entry is set to TRUE, then all v5oneway passwords created or changed after that will be encrypted using the SHA1 algorithm. However, v5oneway passwords created *before* setting this entry to TRUE will use MD5 encryption. These passwords will still be handled correctly by the system; they just will not be encrypted using SHA1. In order to change existing passwords to use the SHA1 algorithm, operators (or the administrator) must manually resave each existing password instance (each Operator ID, or each property of type *password)*.

Available since: Version 6.1

v5portable

Type: boolean

Default: True

Functionality: Version 4 has a simple obfuscation algorithm (MD5) which is used in conjunction with some of the system; customers may be concerned that if a password is encrypted in the same way each time, it is subject to a dictionary attack (which checks well-known passwords). If this entry is set to FALSE, this V4 obfuscation will be used to encrypt passwords.

If this entry is set to TRUE, then a 128-bit AES cipher will be used; a password that is encrypted multiple times in this setup will result in different encrypted strings.

NOTES:

- The default for this setting was changed to TRUE in Version 5.5.
- If this value is set to TRUE, then the encrypted values stored in the Data-Admin-Operator-ID record (and any other place that uses the encrypted password) will not be readable by the version 4 PegaRULES engine. Therefore, this value may only be TRUE after all of the systems sharing the PegaRULES database have been upgraded to V5; this entry will not work in a mixed-version setup (compatibility mode). If both a 4.2 and a 5.1 system will use the same database, this entry MUST be set to FALSE.

Available since: Version 5.1

database Category

The database category contains all database configuration information. To support multiple databases, there can be multiple database URLs, user names, and passwords.

Important: Some of the settings in this category *must* be in the prconfig.xml file, to allow the system to find the database and start up. Some of these settings could be stored either in the prconfig.xml file or the database (as Dynamic System Settings). To prevent confusion, we recommend that you store all the *database* entries in the prconfig.xml file.

batchUpdates

Type: integer

Default: 0

Functionality: This entry controls both whether the batch update functionality will be used, and how many database updates may be done in a batch.

- If the value = 0, the batch functionality is disabled, and the prior-to-5.3 implementation (executeupdate) will be used; updates will be done one-at-a-time.
- If the value = 1, then the new batch functionality will be used, but the end result will be the same, as only one update will be done at a time.

This functionality is shipped disabled. For customers who wish to use this functionality, the recommended value is 10.

NOTE: Due to bugs in both MSSQL code and Oracle code (not in PegaRULES code), this functionality is only recommended for use with DB2 databases.

Available since: Version 5.3

CloseIdleConnections

Type: boolean

Default: true

Functionality: When set to *true,* this setting enables the system to close unused connections to the database. This entry works with the IdleConnectionTimeout setting.

Available since: Version 4.1

drivers

Valid Values: "com.microsoft.jdbc.sqlserver.SQLServerDriver"
"oracle.jdbc.driver.OracleDriver"
"COM.ibm.db2.jdbc.app.DB2Driver"

Functionality: This entry must list all the JDBC drivers that are necessary to connect to any configured databases. The driver names must contain the complete package specifications, and must be separated by semicolons. These classes must be visible to the server – how this is done will vary by server type.

Available since: Version 4.1

dumpStackTrace

Type: string

Default: *blank*

Functionality: When DBTrace is enabled through the prconfig.xml file (using the *dumpStats* entry), this setting turns on the stack trace in DBTrace for the specified events. The value of this entry is a semicolon-delimited list of event types.

NOTE: *dumpStats* must be set to true for this setting to be enabled.

Example:

```
        <env name="database/dumpStats" value="true" />
        <env name="database/traceEvents"
value=”cacheHit;cachedNotFound;cacheMiss" />
        <env name="database/dumpStackTrace" value="cacheMiss" />
```

Valid Values: The event types and their names are:

Event name	Event description
preparedStatementQuery	Execute a prepared statement query
preparedStatementUpdate	Execute a prepared statement update
preparedStatement	Execute a prepared statement
databaseMetadata	Get database metadata
commit	Commit
rollback	Roll back
assignToThread	Connection assigned to a Thread
cacheHit	A cache hit
cachedNotFound	A cached 'not found' hit
cacheMiss	A cache miss
listResultCount	A list that returned results
startActivity	Start of an activity
endActivity	End of an activity
activateConnection	DataSource is activated, obtaining connection
deactivateConnection	DataSource is deactivated, connection returned to JDBC Connection
createDatabaseConnection	DatabaseConnection object created by factory
reuseDatabaseConnection	DatabaseConnection object reused by factory
returnDatabaseConnection	DatabaseConnection object returned to factory
readBlob	Read a blob
writeBlob	Write a blob
runList	Run a list
custom	A custom user-defined operation

Available since: Version 4.1

dumpStats

Type: boolean

Default: false

Functionality: When set to *true,* this setting is used to continually run DBTrace for the entire system (not just when turned on through PAL) and to generate a system-wide database trace file that ends up being written to the ServiceExport directory. The file name starts with "dbOperations" and is followed by a timestamp (example: "dbOperations_20060126T172956_473_GMT.txt")

IMPORTANT: This setting should be used only in very targeted ways, as enabling it will cause performance slowdowns and large amounts of file output. For normal troubleshooting, DBTrace should be enabled through the requestor setting in PAL. This setting aids in tracking any system-wide problems; it shouldn't be used often, and should definitely be turned off after the problem has been traced.

Other prconfig.xml entries (traceEvents and dumpStackTrace) provide finer control over this functionality.

Available since: Version 4.1

historySize

Type: integer

Default: 10

Valid Values: 5 to 100

Functionality: This setting holds the size of history list which is maintained for a database connection (this entry records database actions on a connection)

Available since: Version 4.1

IdleConnectionTimeout

Type: integer

Default: 300

Functionality: When the *CloseIdleConnections* entry is set to true, this entry specifies the amount of time that a connection to a database may remain unused before being considered “idle” and being automatically closed.

Available since: Version 4.2

LockAttemptDelayMS

Type: integer

Default: 250

Valid Values: 100 to 5000

Functionality: When trying to open a record in the database, the system will make some number of attempts (the *MaxLockAttempts* entry) before returning the message that the record is locked. This entry specifies how many milliseconds to wait in between each attempt.

Available since: Version 4.2

MaxLockAttempts

Type: integer

Default: 3

Valid Values: 1 to 100

Functionality: This entry specifies the number of attempts made to open a record in the database before returning a "record is locked" message to the user.

Available since: Version 4.2

storageVersion

Type: string

Default: 6

Valid Values: 4, “iso8859”, “iso-8859”
5, “unicode”, “utf16”, “utf-16”
6

Functionality: Setting this entry to “4” causes the system to store "blob" data using ISO-8859-1 encoding and will prevent the use of non-Latin characters within PegaRULES. StorageVersion 4 writes blobs that are compatible with PegaRULES 03-02.

Setting this entry to "5" enables Unicode support in the database, and writes blobs that are compatible with PegaRULES 4.1.

Setting this entry to "6" not only enables Unicode support, but also enables performance enhancements in the way the data is stored in the database. Storage streams written in this format are compatible with Process Commander Version 4.2.

Available since: Version 4.1

traceEvents

Type: string

Default: *blank*

Functionality: When DBTrace is enabled through the prconfig.xml file (using the dumpStats entry), this setting delineates which specific events should be traced. The value of this entry is a semicolon-delimited list of event types.

NOTES:

- *dumpStats* must be set to true for this setting to be enabled.
- If this setting is not included in the prconfig.xml file, all events are traced by default.

Example:

```
        <env name="database/dumpStats" value="true" />
        <env name="database/traceEvents"
value=”cacheHit;cachedNotFound;cacheMiss" />
        <env name="database/dumpStackTrace" value="cacheMiss" />
```

Valid Values: The event types and their names are:

Event name	Event description
preparedStatementQuery	Execute a prepared statement query
preparedStatementUpdate	Execute a prepared statement update
preparedStatement	Execute a prepared statement
databaseMetadata	Get database metadata
commit	Commit
rollback	Roll back
assignToThread	Connection assigned to a Thread
cacheHit	A cache hit
cachedNotFound	A cached 'not found' hit
cacheMiss	A cache miss
listResultCount	A list that returned results
startActivity	Start of an activity
endActivity	End of an activity
activateConnection	DataSource is activated, obtaining connection
deactivateConnection	DataSource is deactivated, connection returned to JDBC Connection
createDatabaseConnection	DatabaseConnection object created by factory
reuseDatabaseConnection	DatabaseConnection object reused by factory
returnDatabaseConnection	DatabaseConnection object returned to factory
readBlob	Read a blob
writeBlob	Write a blob
runList	Run a list
custom	A custom user-defined operation

Available since: Version 4.1

transactionalLockManagement

Type: string

Default: "advanced"

Valid Values: "advanced" | "standard"

Functionality: When this entry is set to ADVANCED, a class instance must be locked in order to:

- perform an immediate save on the instance (i.e. do an Obj-Save with the *immediate* flag checked);
- perform an immediate delete on the instance (i.e. do an Obj-Delete with the *immediate* flag checked);
- perform a deferred save (i.e. an Obj-Save with the *immediate* flag unchecked) on the instance and commit it;
- perform a deferred delete on the instance and commit it.

If an instance is not locked in any of the above cases, an error will occur. In the case of deferred operations, the error will occur at commit time. There are some exceptions:

- Locking is not required on instances of classes where locking is not enabled.
- If saving a brand new instance (that is, an instance that did not come from the database), it does not need to be locked.

When this entry is set to "standard", then locking is not required to save or delete instances.

NOTE: Running in Advanced mode is a PegaRULES Process Commander "best practice". However, some customers may choose "standard", to run in backward compatibility mode and allow configurations that require relaxed transactional design principles.

Available since: Version 4.2 SP2 "SmartBuild"

baseTable Subcategory

The baseTable category is a subcategory of the database category. It is explicitly set when a fully qualified table name is defined during deployment.

catalog

Type: string

Default: *blank*

Example: "engineering"

Functionality: the catalog name associated with base table

Available since: Version 4.1

name

Type: string

Default: "pr4_base"

Functionality: the name for base table for system startup information.

Available since: Version 4.1

schema

Type: string

Default: *blank*

Example: "genetics"

Functionality: Specifies the database table name in which to store instances of the given class. This table must already be defined in the database schema.

Available since: Version 4.1

databases subcategory

databases is a subcategory of the *database* category. Under this subcategory, the prconfig.xml file contains one subcategory for the "bootstrap" (starting) database, and may also contain additional subcategories for other databases used by the system. The name of each of these categories must be the same as the name of the database – the "logical" database name. This logical name does not have to be the same as the physical database name, but must match the database names in the Data-Admin-DB-Name and Data-Admin-DB-Table rules.

NOTES:

- A subcategory for "PegaRULES" is required for all systems. Others may be added, if necessary, for additional customer-defined databases.
- For all systems using the "PegaRULES" subcategory, these settings *must* be in the prconfig.xml file – otherwise, the system won't be able to start.
- For systems which have additional customer-defined databases and are using these settings to point to those databases, the settings may be located either in the prconfig.xml file or in the database as Dynamic System Settings.

IMPORTANT: While the **databases** section *can* be used to define connection configuration to all your databases, it is highly recommended that only the PegaRULES bootstrap information be maintained in this file. Other connection information should be defined only as Data-Admin-DB-Name class instances in Process Commander. Centralizing the database configurations in these class instances (rather than the prconfig.xml file) minimizes changes to the prconfig.xml, reduces the need for complex configuration migration schemes, and provides a useful construct for testing connection configuration without requiring a system restart.

Example of subcategory levels for the **database** category:

```
<!-- database settings -->
      <env name="database/databases/pegarules/datasource"
value="java:comp/env/jdbc/PegaRULES" />
<env name="database/databases/pegarules/maxconnections" value="10" />
```

The **databases** subcategory is used to define the database connection configuration to bootstrap PegaRULES. There are two common setups for connecting to the PegaRULES database:

- Process Commander connection pooling using JDBC drivers
- JDBC DataSource Support

1) JDBC DataSource

This is the preferred mechanism. During the installation process, the JNDI name of a local resource reference to a JDBC DataSource is defined, and then the application server is used to manage the JDBC configuration itself. A JDBC DataSource must be created in the application server and then bound to the local resource reference in your PegaRULES deployment.

PegaRULES comes prepackaged with a local resource reference, so that the DataSource may be immediately defined in the application server, and PegaRULES may be started. Thus, in most cases, the prconfig.xml will have just the following line:

```
<env name="database/databases/pegarules/datasource"
value="java:comp/env/jdbc/PegaRULES"/>
```

(Please consult your application server's documentation for full details on this setup.)

2) PegaRULES-Managed Connections

For deployments where J2EE isn't available, or its use is not desired, PegaRULES also supports managing JDBC Connections directly. In these cases, additional information must be provided to PegaRULES: a JDBC Driver class, Connection URL, and credential information. In most cases, the following configuration is sufficent:

```
<env name="database/drivers" value="com.your-rdbms.provider.jdbc.Driver"/>
<env name="database/databases/pegarules/url" value="jdbc:url://for-your-
database"/>
<env name="database/databases/pegarules/username" value="your_db_user"/>
<env name="database/databases/pegarules/password" value="your_db_password"/>
```

(Please consult your JDBC driver documentation for information on the class name and URL format for connecting to your database.)

Occasionally, databases drivers need a more flexible configuration scheme to allow arbitrary connection configuration to be passed to the JDBC driver. In these cases, a Java-style property may be used to define these entries, which may then be referenced in your configuration:

```
<env name="database/drivers" value="com.your-rdbms.provider.jdbc.Driver"/>
<env name="database/databases/pegarules/url" value="jdbc:url://for-your-
database"/>
<env name="database/databases/pegarules/propertiesFile"
value="/opt/your_config/config.properties"/>
```

For this style of configuration, the username and password will be contained within the properties file. (Please consult the documentation of your JDBC provider for how to specify these values, and what additional properties are available to the connection configuration.)

adminPassword

Type: string

Functionality: Beginning in Version 5.4, the Install Guide recommends that the database UserID be set up with CREATE TABLE and ALTER TABLE privileges. If the DBA does not want these privileges on the default database userid, the Admin Database User may be set up with these privileges.

For a Pega Connection Pooling setup, if the adminUserName and adminPassword entries are in the prconfig.xml file, the system will automatically use this connection for the features requiring create or alter privileges. (See KB #25318, *How to restrict default database privileges*, for full details.)

This entry holds the password for the Admin Database User.

Available since: Version 5.4

adminUserName

Type: string

Functionality: Beginning in Version 5.4, the Install Guide recommends that the database UserID be set up with CREATE TABLE and ALTER TABLE privileges. If the DBA does not want these privileges on the default database userid, the Admin Database User may be set up with these privileges.

For a Pega Connection Pooling setup, if the adminUserName and adminPassword entries are in the prconfig.xml file, the system will automatically use this connection for the features requiring create or alter privileges. (See KB #25318, *How to restrict default database privileges*, for full details.)

This entry holds the name of the Admin Database User.

Available since: Version 5.4

databaseType

Type: string

Valid Values: "MS SQLServer"
"Oracle"
"DB2"
"DB2/OS/390"
"Apache Derby"

Functionality: SQL statements differ for different databases, so it is necessary to know the database type in order to choose the correct DDL (code) to run to create or alter tables in the database. If the system is using a driver for the database connection that is from the actual vendor, then that driver will provide the correct information on the database type. If the driver is a third-party driver, however, there may be issues getting the correct database type. In this situation, use this entry to specify the appropriate database type.

Available since: Version 5.4

dataSource

Type: string

Example: "your/ds/name"

Functionality: This setting holds the string which is a JNDI reference to a JDBC DataSource object. (This is the "JNDI name" that is specified in the application server configuration.)

Available since: Version 4.2

datasourceAdmin

Type: string

Example: "your/ds/name"

Functionality: Beginning in Version 5.4, the Install Guide recommends that the database UserID be set up with CREATE TABLE and ALTER TABLE privileges. If the DBA does not want these privileges on the default database userid, the Admin Database User may be set up with these privileges.

For a datasource setup, if this entry is in the prconfig.xml file, the system will automatically use this connection for the features requiring create or alter privileges. (See KB #25318, *How to restrict default database privileges*, for full details.)

This entry holds the name of the datasource which points to the Admin Database User.

Available since: Version 5.4

maxConnections

Type: integer

Example: "5"

Functionality: This setting allows the developer to limit the number of PegaRULES connections to the database. For example, setting the value for this entry to "5" will limit PegaRULES to 5 connections for the associated databases.

Note that this entry limits the number of connections that *PegaRULES* may make; it is also possible to set a value in WebSphere or WebLogic to limit the number of connections the *application server* will allow. (Please reference PDN # 17232 – *How To: Set the Maximum Number of Database Connections* for full details on both these settings, and how they affect PegaRULES differently.)

WARNINGS:

- Setting this value too low may result in deadlocks.
- Setting this value higher than the maximum connections supported by your database configuration may result in a request hanging while waiting for a connection that never becomes available from the database (JDBC drivers will typically wait for a connection and not report an error when a connection cannot be obtained).

Available since: Version 4.2

maxConnectionsTimeout

Type: integer

Default: 10

Functionality: This setting is used in conjunction with the *maxConnections* setting, and displays the maximum amount of time, in seconds, that a Java thread will wait for a connection if the pool of connectors is empty. If this amount of time passes, and no connection is available, a Database Exception error will occur.

Available since: Version 5.1

minConnections

Type: integer

Default: 0

Functionality: This setting defines the minimum number of connections to create for the connection pool for a particular database (Data-Admin-DB-Name instance). This setting specifies that when the system starts up and connects to the database, whether it should specify zero or some number of connections to the database.

Available since: Version 5.3

password

Type: string

Functionality: This entry holds the password used to connect to the database (which may be stored in a separate encrypted file if so desired. See the Encryption Tech Note for details).

Available since: Version 4.1

propertiesFile

Type: string

Example: "f:/weblogic/pegarules/prweb/WEB-INF/classes/oracle.conf"

Functionality: This setting holds the path to the *oracle.conf* file and would be added in with the url, name, & password information. (Please reference PDN # 16433 – *FAQ: Oracle 9i Client Versions* for full details on how and why to set up the *oracle.conf* file.)

Available since: Version 4.2

reservedConnections

Type: integer

Default: 1

Functionality: This entry is used in conjunction with the *maxConnections* setting, and holds the number of connections which are reserved for "critical" operations, such as classmap information.

Available since: Version 5.1

url

Type: string

Example: "oracle.jdbc.OracleDriver;com.microsoft.jdbc.sqlserver.SQLServerDriver"

Functionality: This entry holds the JDBC URL used to connect to the database.

Available since: Version 4.1

userName

Type: string

Functionality: This entry specifies the user name used to connect to the database.

Available since: Version 4.1

encoding subcategory

ASCII characters are one byte in size. Unicode handles extra characters not available in ASCII, and so characters in a Unicode database (UTF-8, for example) could be 1, 2, or 4 bytes per character. This can cause issues with PRPC database table column size. A column may be set to be “64 characters”, and the system equates that to 64 bytes. If an entry is made of 60 characters, that will fit in ASCII, but in Unicode, that could be 60, 120, or 240 bytes; the second and third options will overflow that column width, resulting in errors.

Therefore, when Database Encoding is enabled, the system divides all column widths by the database charset value to accommodate the Unicode byte sizes:

- for MSSQL, the width is divided by 2 (so a 64-byte column would allow 32 characters)
- for Oracle and DB2 databases, the width is divided by 4 (so the 64-byte column would allow 16 characters).
-

useDatabaseEncoding

Type: boolean

Default: False

Functionality: This setting enables or disables the use of Database Encoding. When this entry is set to *true,* the system will divide the column width by the database charset value.

Available since: Version 6.1

MaxEncodedCharSize

Type: integer

Default: *the charset value of the database*

Valid values: 2, 3, 4

Functionality: Some languages only use two bytes in Unicode (such as Western European languages); others (such as Chinese) can use four bytes for their characters. If the number of bytes required for each character in a language is different than the database charset value, this entry may be set to indicate what factor to use.

NOTE: For this value to be read, *useDatabaseEncoding* must be enabled (set to *true*).

Available since: Version 6.1

DeclarePages Category

This category contains settings relating to the Declare Pages functionality.

DefaultIdleTimeSeconds

Type: integer

Default: 86400

Functionality: The system periodically checks the global shared clipboard pages collection to see if there are any pages which haven't been used in a long time. This entry specifies the amount of time (in seconds) between checks. If a declarative page hasn't been used within that time limit, it will be removed from the node. If it is needed again, it will be automatically constructed (as usual).

NOTE: Setting this value too low may cause slower performance; the Declare Pages may be "expired" and deleted too quickly, forcing the system to constantly run the activity to recreate them.

This time between checks may also be specified in the Declare Pages rule form; any value specified there will override this entry.

Available since: Version 5.3

MemoryUsagePercentLimit

Type: integer

Default: 5

Functionality: This setting shows what percentage of total JVM memory the Node-scope Declare Pages functionality is using. The default is 5%; if the JVM were set to 600MB, this limit would thus be 30MB. If this limit is exceeded, then an alert is put into the PegaALERT log.

Available since: Version 5.3

fua Category

This category contains settings which are used to configure and manage the
When Process Commander loads rules for the first time, both at startup and when a rule is changed and must be loaded again, it generates Java source files and then compiles the source files into Java class files. This file generation process is referred to as Rules Assembly (in previous releases, called First-Use Assembly or FUA).

The fua category contains settings that determine the size of the rule caches. Process Commander maintains a global cache for the system, as well as individual personal caches for each user that has rule checkout enabled in their operator ID records.

class subcategory

instanceCountLimit

Type: integer

Default: 20000

Functionality: This entry holds initial allocation size of the Assembled Class Entries in the l Rules Assembly Instance Cache. The optimal size for each cache will vary based on the customer's application and rules use, and should be tuned by the application developer for maximum performance efficiency.

Available since: Version 5.5

global subcategory

instanceCountLimit

Type: integer

Default: 20000

Functionality: This entry holds initial allocation size of the Global Rules Assembly Instance Cache. The optimal size for each cache will vary based on the customer's application and rules use, and should be tuned by the application developer for maximum performance efficiency.

Available since: Version 4.1

personal subcategory

instanceCountLimit

Type: integer

Default: 20000

Functionality: This entry holds initial allocation size of the Personal Rules Assembly Instance Cache. The optimal size for each cache will vary based on the customer's application and rules use, and should be tuned by the application developer for maximum performance efficiency.

Available since: Version 4.1

shared subcategory

instanceCountLimit

Type: integer

Default: 20000

Functionality: This entry holds initial allocation size of the Shared Rules Assembly Instance Cache. The optimal size for each cache will vary based on the customer's application and rules use, and should be tuned by the application developer for maximum performance efficiency.

Available since: Version 6.1

GarbageCollection Category

This category contains settings which provide information about the garbage collection functions of the system.

Logfile

Type: string

Example: "C:\pegarules\logs"

Functionality: This entry sets the location of where the VerboseGC output is located for this particular PegaRULES node. This value is used by the System Management Application to display GC statistics. NOTE: This value must match the location supplied to the JVM in conjunction with the *verbose gc* option.

Available since: Version 4.1

HTTP Category

This category contains settings which customize the behavior of interactive requests between the client machine and the PegaRULES server.

DefaultCachingTimeout

Type: integer

Default: 86400

Functionality: The max-age element of the CACHECONTROL setting in HTTP commands controls how long content (in this case, static content such as the login image) is considered by the browser to be "current." If content is not "current," then the browser will request updated data from the server.

This setting holds the maximum age that the static content is considered "current." The default is 24 hours (86,400 seconds).

Available since: Version 5.1

serverGZIPEnabled

Type: boolean

Default: True

Functionality: This setting enables GZIP compression of static content that will be sent to an HTTP client.

NOTE: Prior to 5.1, this setting was in Initialization/GZIPStaticContent

Available since: Version 5.1

serverZIPEnabled

Type: boolean

Default: False

Functionality: This setting enables ZIP compression of static content that will be sent to an HTTP client.

NOTE: Prior to 5.1, this setting was in Initialization/ZIPStaticContent

Available since: Version 5.1

SetSecureCookie

Type: boolean

Default: False

Functionality: This setting indicates whether to set the "secure" flag on the cookie itself. If this flag is set, then the cookie will only be sent from the client to the server if the browser is using a secure protocol – i.e., using either **https** or **SSL**. If the browser is *not* using a secure protocol, then the cookie is not sent.

If the server does not receive a cookie with a request, it assumes that this user does not have a requestor and sends back the logon screen. Note that even if the user re-logs on, if there is no cookie included in any of the browser requests (due to not using either https or SSL), the user will continue to only get the login screen as a response.

This security is provided in case a hacker uses a "sniffer" program on the network line. If this entry is set to *false* (i.e., not secure), then the "sniffer" will display the cookie requestor information in clear text. If the entry is set to *true,* then the data in the cookie will be obfuscated.

Available since: Version 5.3 SP1

ShrinkJavaScriptEnabled

Type: boolean

Default: true

Functionality: This setting enables compression of the Java static content files (from Rule-File-Text) when downloading them from the server to client machines. Some of the effects of the shrinking include:

- spaces are removed
- comments are removed
- variable names within functions are obfuscated (e.g. "oMyVariable" would be changed to something like "_a")

NOTES:

- If ShrinkJavaScriptEnabled is set to true (i.e., JavaScript file shrinking is enabled), then developers can use a setting called "Disable Compacting" in the user's Desktop Preferences (General tab) to override this entry and turn the file shrinking off.
- If ShrinkJavaScriptEnabled is set to false (i.e., JavaScript file shrinking is disabled), then the Disable Compacting setting will have no effect.

Available since: Version 5.4

StaticContentCommitAll

Type: boolean

Default: false

Functionality: Setting this value to TRUE allows Static Content Files to function properly when using WebSphere 6.0. If this setting is not enabled for systems running WebSphere 6.0, the system will autodetect that WS 6.0 is present and automatically enable the setting. (Other application servers, such as WebLogic, will not have this setting turned on.)

If this setting is specified in the prconfig.xml file, the specified setting will override the autodetection, and whatever is specified will be the behavior.

Available since: Version 4.2 SP2 "SmartBuild"

UseNoCacheHeaders

Type: boolean

Default: False

Functionality: Process Commander sends both static content (such as images or the login screen) and dynamic content (such as work item data) to the user's browser on the client machine. Static content is cached on the client machine, and set to expire after 24 hours, unless that content on the server has changed. (So if a user has a customer logo as part of a work item, for example, and that logo does not change, it doesn't have to be sent down to the client every time a new work item is opened – only once every 24 hours.)

Dynamic content is also cached on the client machine; however, it is set to expire immediately. If the user tries to access that content again, it will automatically be brought down from the server again. This technique provides maximum compatibility with existing browsers, but may leave sensitive data on the client computer.

This setting is used to prevent dynamic content from being cached on the client machine. The default for this setting is *false,* meaning that the dynamic content will be cached on the client machine. When this entry is set to *true*, the dynamic content sent to the client will not only be set to expire immediately, but also will not be cached.

NOTE: A side effect of setting this entry to *true* is that the Tracer functionality will be disabled.

Available since: Version 5.3 SP1

StaticContent subcategory

enableCapture

Type: boolean

Default: False

Functionality: When set to true, this entry enables the capture of static content in order to store that data in a server close to end users. For details, see KB #25587.

Available since: Version 5.5

remoteWebServerAccessGroupName

Type: string

Example value: "GRP350App:WorkUsers"

Functionality: This setting indicates the access group name for which static content should be captured.

NOTE: This setting is only valid when the *enableCapture* entry is set to TRUE.

Available since: Version 5.5

Identification Category

The identification category contains definitions related to identifying the system.

KeyringAlgorithm

Type: string

Default: "DESede"

Functionality: This value should be set if a different encryption algorithm is desired to encrypt the PegaRULES keyring file.

NOTE: This setting is used in conjunction with the v4 keyring cipher only.

Available since: Version 4.1

KeyringLength

Type: integer

Default: 168

Functionality: This value should be set if a different key length is desired. Note that the value specified must be compatible with the algorithm selected, and consistent with the security policy files installed in the Java JVM.

NOTE: This setting is used in conjunction with the v4 keyring cipher only.

Available since: Version 4.1

KeyringPrefix

Type: string

Default: "pegarules"

Functionality: This value should be set if more than one PegaRULES system will be deployed in the same application server and JVM.

When trying to locate the pegarules.keyring file, the Java property <KeyringPrefix>.keyring is examined first. If this property is defined, its value is the full file specification of the keyring file to be used. Since each keyring file is unique to a single deployment of PegaRULES, a distinct property is required for each deployment. This option permits you to specify such a unique value. In most cases the default is appropriate.

Available since: Version 4.1

SystemName

Type: string

Default: "pega"

Functionality: Name of instance in Data-Admin-System for this server. Also affects System Pulse; defines the scope of response to messages that are sent out via the pulse. Systems can send messages that will only affect their own system (possibly across multiple machines) or all systems that share a database across all machines.

Available since: Version 4.1

SystemType

Type: string

Valid Values: "emergency"
"standard"

Functionality: The systemType indicates whether this installation of PegaRULES should run as a standard system ("standard") or in "emergency" mode.

In "emergency" mode it is possible to edit the on-disk Java file for assembled rules and have those changes compiled and loaded. This should be used *only* when a serious problem prevents accessing the PegaRULES system in a normal fashion due to bad logic in a rule needed to initialize or log into PegaRULES.

Available since: Version 4.1

Indexing Category

In order to make finding specific rules easier in the PegaRULES Process Commander system, a full-text search has been implemented, based on the Apache Lucene open source indexing software. These settings control the process of indexing the database. Separate index files are used for rules, data instances, and work objects.

In Version 5.1, enhancements were made to the Lucene indexing functionality for multi-node systems. The new functionality allowed the developer to designate *one* node of a Process Commander system to store the index files; all other systems would point to that node, and the entire system would use one index. In this case, all the prconfig.xml settings would be *ignored*; instead, Dynamic System Settings would be used (so all nodes would have the same server information).

If a developer wished to keep the 4.2 behavior of an index on each node, however, they must set the *runCompatibilityMode* entry to TRUE. At that point, the system would continue to use the prconfig.xml settings, and ignore the Dynamic System Settings.

enabled

Type: boolean

Default: true

Functionality: This entry allows globally enabling or disabling rule indexing on a node of the system (the node on which this prconfig.xml file is being used). If the entry is set to True, then changes to rules, work, or data may be indexed via the Lucene functionality (depending upon their individual class settings - ruleEnabled, workEnabled, dataEnabled).

This index will be stored either in the explicitIndexDir (if specified), or written to a directory under the context root: /prweb/PegaRULESIndex.

NOTE: In Version 5.1, this setting will be *ignored* unless *runCompatibilityMode* is set to TRUE.

Available since: Version 4.2 SP2 "SmartBuild"

explicitIndexDir

Type: string

Example: "C:\index"

Functionality: This setting can be used to force PegaRULES to write its index files to a certain location. When not specified, this value defaults to a sub-directory of the PegaRULES temp directory called PegaRULESIndex. The value specified MUST be an absolute file path, and there must be at least 300MB of free space available in the directory. (The index file will grow as changes and additions are made to rules.)

NOTES:

- Index directories should not be shared across distinct PegaRULES installations.
- In Version 5.1, this setting will be *ignored* unless *runCompatibilityMode* is set to TRUE.

Available since: Version 4.2 SP2 "SmartBuild"

dataEnabled

Type: boolean

Default: false

Functionality: This entry enables or disables indexing for Data- classes.

NOTES:

- This entry is only relevant if the "enabled" entry is set to TRUE.
- In Version 5.1, this setting will be *ignored* unless *runCompatibilityMode* is set to TRUE.

Available since: Version 4.2 SP4

ruleEnabled

Type: boolean

Default: true

Functionality: This entry enables or disables indexing for Rule- classes.

NOTES:

- This entry is only relevant if the "enabled" entry is set to TRUE.
- In Version 5.1, this setting will be *ignored* unless *runCompatibilityMode* is set to TRUE.

Available since: Version 4.2 SP2 "SmartBuild"

runCompatibilityMode

Type: boolean

Default: false

Functionality: This setting allows the system to use the Process Commander v4.2 indexing behavior for the Lucene index. When this setting is enabled, the Lucene index will be built and maintained on *each* node in a multi-node system, bypassing the multi-node enhancements made in v5.1 which allow the developer to designate a node in the system as the Lucene index "server."

NOTE: If this setting is enabled, then the Dynamic System Settings for indexing will also be ignored, and the prconfig.xml settings will be used.

Available since: Version 5.1

workEnabled

Type: boolean

Default: false

Functionality: This entry enables or disables indexing for Work- classes.

NOTES:

- This entry is only relevant if the "enabled" entry is set to TRUE.
- In Version 5.1, this setting will be *ignored* unless *runCompatibilityMode* is set to TRUE.

Available since: Version 4.2 SP4

Initialization Category

The Initialization category defines the system environment.

activities

Type: string

Example: "MayStartActivities.xml"

Functionality: This setting holds the name of the .xml file which contains the list of approved activities for an Internet-accessible application.

NOTE: If **nodetype** is set to *web,* then the **activities** setting is required.

For full details on this entire process, please see KB #25186, *Internet Application Composer – How to configure a Process Commander web node*

Available since: Version 5.4

ContextRewriteEnabled

Type: boolean

Default: false

Functionality: This setting enables Reverse Proxy Server functionality for PegaRULES. In order to use either the SetContextURI HTTP Header setting or the SetBaseHTMLcontext entry (Initialization category) in the prconfig.xml file, this entry must be TRUE.

NOTE: If a URL is specified for the **GatewayURL** setting, that URL will override any entries for this and SetBaseHTMLContext.

Available since: Version 4.1

CPUTimerLevel

Type: string

Valid Values: "FULL"
"TOTALSONLY"
"NONE"

Functionality: This entry controls the CPU timing measurements. Systems on a Windows OS default to FULL; other OS's (like UNIX) default to TOTALSONLY - only pxTotalReqCPU and pxProcessCPU are measured. To disable all CPU timing measurements, set to NONE. Information message output is sent to the log file if setting is either TOTALSONLY or NONE.

Available since: Version 6.1

DisableAutoComplete

Type: boolean

Default: false

Functionality: By default, when users first log into the system, Internet Explorer prompts them with a screen asking whether they would like Windows to remember their password. If the user clicks **Yes,** then the next time they log in, as soon as they start to type in their User ID, they will be prompted with a drop-down box containing the user ID they typed in the last time (the *AutoComplete* function). If they choose one of the usernames in the dropdown, the system will also enter their password into the next field. This can be a convenience to users, but also makes it easier to guess or simply use the usernames and passwords on login screens.

This setting enables or disables the AutoComplete function. If this entry is set to *true,* then users will have to type in their usernames and passwords each time without prompting.

.

Available since: Version 5.3 SP1

DisplayExceptionTraceback

Type: boolean

Default: true

Functionality: When an error occurs in Process Commander, the exception screen is displayed. Clicking on the **Show Exception Details** button will show data which includes stack trace messages, which can contain system details that may be useful to users trying to circumvent system security.

Even if the button is not clicked, the stack trace information may be displayed by right-clicking on the browser screen and using the **View Source** option.

This setting controls the display of the stack trace information. When set to *false,* this entry will remove the **Show Exception Details** button from the error screen, and also remove all the stack trace data from the **View Source** screen.

Available since: Version 5.3 SP1

explicitTempDir

Type: string

Example: "C:\pegarules\temp"

Functionality: This setting specifies the full path which should be used as the work directory. This allows the developer to use a shorter path to the temp (work) directory, in order to avoid exceeding various platform path name limits. The default continues to be the work directory supplied by the application server.

Important Notes:

- Specifying the *explicittempdir* setting in the prconfig.xml file but leaving the value blank will cause the system to create the temp (work) directory in the default location supplied by the application server. *A value must be specified here in order for the temp path to be shortened.*
- If this entry is used, then the useJavaIoTempDir entry must be set to FALSE.

- The value specified by this entry will be ignored unless the directory already exists. Make sure that the user has created the directory specified prior to starting PegaRULES. If the directory does NOT exist, then the temp directory provided by the application server will be used and no warning or error will be logged.

- It is possible to use Java property substitutions to specify this setting as a JVM system variable (example: ${*setting*}). In WebSphere, the shipped WAS policy file expects a Java property named **pega.tmpdir** to point to the explicit temp directory location, so using this property name in conjunction with the substitution (e.g., ${pega.tmpdir}) would keep the reference consistent.

Available since: Version 4.2

extractLibraries

Type: string

Default: "false"

Valid Values: "true"
"false"
"compile"

Functionality: By default, the Rule-Utility-Libraries are extracted once, the first time the system is started up.

If this setting is set to "true," then the libraries will be extracted every time the PegaRULES instance is started on the server. (This may add several minutes to the startup time of the PegaRULES instance.)

If this setting is set to "false," the libraries will *only* be extracted when either the PRGenJava files are empty, or the PegaRULES_Extract_Marker.txt file is missing.

Available since: Version 4.1

GatewayURL

Type: string

Example: "http://support.acme.com:6020/prgateway/PRPCGateway/SupportOnLine"

Functionality: This entry is required when using the Pega Composite Gateway functionality, and is only relevant for that feature. The value is comprised of:

- **protocol** - http or https
- **host** – full domain name for accessing the gateway
- **port** – port number for accessing the gateway
- **gateway servlet context** – gateway servlet context root and servlet name
- **logical host name** – a value that must match the logical host name specified in the div element where PRPC is embedded in an external web portal application

NOTE: If a URL is specified for this setting, that URL will override any entries for the initialization settings **SetBaseHTMLContext** and **ContextRewriteEnabled**.

For full details, see KB #25187: *Internet Application Composer: How to set the gatewayURL configuration parameter.*

Available since: Version 5.4

httpcompression

Type: boolean

Default: True

Functionality: This setting enables use of compression when sending data to an HTTP client.

Available since: Version 4.1

HTTPLockRetryInterval

Type: integer

Default: 20000

Functionality: Before any HTTP operation executes in PRServlet, exclusive access to the requestor object is required. This entry holds the amount of time, in milliseconds, the system will delay between each requestor lock attempt.

Available since: Version 4.2

HTTPMaxLockAttempts

Type: integer

Default: 6

Functionality: Before any HTTP operation executes in PRServlet, exclusive access to the requestor object is required. This setting control how many attempts are made to get that exclusive access to the requestor.

Available since: Version 4.2

MaximumFileUploadSizeMB

Type: integer

Default: 25

Functionality: This entry holds the size, in MB, of the largest file that can be uploaded into the PegaRULES database. NOTE: Files larger than 25MB are not recommended for uploading into the database. Files of this size put a strain on the system, and increase the chance that the server will spend all its time loading and unloading files (making the system look like it's under a denial-of-service attack).

If a user attempts to upload a file larger than this setting, the following error will occur:

"HTTP Status 413 – User attempted to upload a file larger than 25 MB. Please contact the System Administrator."

Available since: Version 4.1

NodeClassification

Type: string

Example: "Agents"

Functionality: This setting creates a "classification" of a node type, so that all the nodes of this type use these configuration settings.

IMPORTANT: As this setting is one of the "keys" of the Dynamic System Settings in Version 6.2, this setting *must* be in the prconfig.xml file.

Available since: Version 6.1 SP2

nodetype

Type: string

Default: "intranet"

Valid Values: "intranet"
"webtesting"
"web"

Functionality: When setting up an Internet Application in Process Commander, the overall procedure is to first write the application in a standard PRPC setup ("intranet" mode); then exercise that application in "webtesting" mode to obtain a list of the activities which may be run by users from the Internet; and finally, set up the application to be accessed by users on the Internet ("web" mode).

This setting holds the node type for the application during this process:

intranet – standard internal deployment of a PRPC application

webtesting – a web application in the testing phase. The *webtesting* nodetype records all the activities run by the developer into a Log-Security-WebNode record, to determine what was run and what is allowed or not allowed.

web – web application, which is accessible from the Internet, with a gateway in front of it

NOTE: If nodetype = *web,* the **activities** entry must be set as well.

For full details on this entire process, please see KB #25186, *Internet Application Composer – How to configure a Process Commander web node*

Available since: Version 5.4

ProfileApplication

Type: boolean

Default: false

Functionality: Changing this setting to *true* will enable the Performance Profiler. The Performance Profiler is a tracing tool introduced in Process Commander Version 4.2, and should be used in conjunction with PAL. Enabling the Profiler setting will cause a comma-delimited text file to be written to the PegaRULES temp directory, which can be opened as a .csv file, or opened in Excel. These files are named by the requestor identifier and the thread name:

```
HCA3E5D0AFD332938A7CF63DB0A0E73B6_STANDARD.csv
```

IMPORTANT: The Performance Profiler should *not* be left running in a production or even a development system! Enabling the Profiler will print profile data on *all* threads running in the system, which will have a noticeable impact on system performance. **The Performance Profiler should *only* be enabled for troubleshooting, and should then be immediately disabled**.

Available since: Version 4.2

PromoteEmbeddedPortals

Type: boolean

Default: False

Functionality: The Process Commander login image displays at the top of the frame. However, it is possible for a malicious user to configure HTML to have a frameset which "frames the frame" and encloses the PRPC login image. The malicious user could then write a script at the very top of their outer frame which "listens" to keystrokes, and records them for future use. There is no visible way to detect this setup; the login screen would look the same.

This setting prevents this situation. When set to *true,* this entry verifies that the Process Commander login image is at the very top of the HTML page, and not embedded in any other frames.

Available since: Version 5.3 SP1

pxServiceExportPath

Type: string

Functionality: This entry holds the location of the directory in which exported EJBs and other Rule-Service-* related artifacts should be stored.

Available since: Version 4.1

pxTextDownloadEncoding

Type: string

Default: "UTF-8"

Functionality: This entry holds the type of character encoding used for text files downloaded to the client. Only the use of UTF encodings will permit full multi-language support (for mixed encodings on a single page).

Available since: Version 4.1

pxTextExtractEncoding

Type: string

Default: “UTF-8”

Functionality: This entry holds the type of character encoding used for text files extracted from the database for use by the client. Only the use of UTF encodings will permit full multi-language support (for mixed encodings on a single page).

Available since: Version 4.1

pxTextUploadEncoding

Type: string

Default: platform default encoding, except that "Cp1047" is changed to "ISO-8859-1"

Functionality: This entry holds the type of character encoding assumed for uploaded files from the client if no UTF signature or byte order mark is detected. Note that if the platform's default encoding is EBCDIC, uploaded files are assumed to be encoded in ISO-8859-1 (if no UTF encoding could be detected).

Available since: Version 4.1

RedirectOnTimeout

Type: boolean

Default: False

Functionality: When configured, the Connection Timeout feature will launch a confirmation dialog box which warns the user that their session will expire soon. This feature also allows the user to send te response from timeout processing to a modal dialog box or a new window. (See the *Configuring Timeouts* article on the PDN for details.) This feature will not, however, block the user’s work items – possibly containing sensitive data – from appearing on the screen behind the timeout warning or dialog box.

This setting controls the display of the user screen in a timeout situation. When this entry is set to *true,* the warning message would still be displayed before timeout; however, once the timeout is reached, the user will be logged out of Process Commander and the screen will display the "logged out" image. In order to keep working, the user must log into PRPC again. This prevents the sensitive data from being visible when users are no longer at their desk.

Available since: Version 5.3 SP1

SetBaseHTMLContext

Type: string

Example: "http://revproxy/AcmeCorp/setup"

Functionality: This entry enables Reverse Proxy Server functionality for PegaRULES.

NOTES:

- In order to use this setting, the **ContextRewriteEnabled** setting must be set to *true.*
- If a URL is specified for the **GatewayURL** setting, that URL will override any entries for this and **ContextRewriteEnabled**.

Available since: Version 4.1

settingsource

Type: string

Default: "Merged" (6.2 default)
"File" (6.1 SP2 default)

Functionality: If this entry is set to *merged,* the system will check for settings in the database (Data-Admin-System-Settings), as well as the prconfig.xml file. If settings are found in both the file and the database, *the file setting will take precedence.*

If this entry is set to *file*, the system will *ignore* Data-Admin-System-Settings as entries, and use *only* the entries in the prconfig.xml file.

IMPORTANT: As this setting tells the system where to look for the rest of the entries, this setting *must* be in the prconfig.xml file.

Available since: Version 6.1 SP2

SubmitObfuscatedURL

Type: string

Default: "optional"

Valid Values: "optional"
"required"
"log"

Functionality: This setting determines whether the server will accept URLs with query strings which are in clear text, or whether they *must* be obfuscated. There are several valid values:

- **required** – The server will *only* accept URLS with obfuscated query strings. If a query string is in clear text, that request will be rejected, and an HTTP 404 (Bad Request) response will be sent back to the client.
- **optional** – The server will accept URLS with *either* clear-text or obfuscated strings.
- **log** - The server will accept URLS with *either* clear-text or obfuscated strings, and the clear-text URLs *only* will be written to the console log. (Obfuscated URLs will not be logged.)

IMPORTANT NOTES:

- This setting will only work when *urlencryption* is turned on; otherwise, the URL query strings cannot be obfuscated.
- If *urlencryption* is enabled, then it is recommended to use the *optional* or *log* settings ***for debugging or testing only.*** Do not use them in production – the console log will fill up with URL information.

Available since: Version 5.3 SP1

TracerMaxWait

Type: integer

Default: 3600

Functionality: maximum number of seconds a requestor can be stopped at a break or watch point before automatically resuming. Defaults to 1 hour.

Available since: Version 4.2 SP2 "SmartBuild"

urldebug

Type: string

Default: "none"

Valid Values: "none"
"in"
"out"
"both"

Functionality: This setting specifies which of the URLs with obfuscated query strings should be written to the *urlrrwriter* log file. (See next setting for file information.)

- **none** – None of the entries will be written to the log.
- **in** – The requests sent from the browser to the server will be logged. The log entry for each inbound request will show what the server has received as an obfuscated request, and then the associated unencrypted request.
- **out** – The responses sent from the server to the browser will be logged. The log entry for each outbound request will show the unencrypted output stream, and then the obfuscated response.
- **both** – Both types of entries will be written to the log.

NOTE: This setting should be considered a tool for testing and debugging *only*. Enabling logging – especially the *both* value – creates a great deal of output. This setting should be disabled for production systems.

Available since: Version 5.3 SP1

urlencryption

Type: boolean

Default: True (*originally false; true as of 6.1)*

Functionality: This setting enables or disables the obfuscation of the URLS which are sent back to the client. If this entry is set to *false,* then no obfuscation occurs.

If this entry is set to *true*, then before sending a response back to the client, the system will scan the output stream for two types of query strings:

- "pyActivity=XXX"
- "pyStream=XXX"

If either of these are found, the system will extract them and replace them with obfuscated values.

NOTE: If this setting is changed from FALSE to TRUE, but testing does not show that URL encryption has been enabled, clear the browser cache and retest.

Available since: Version 5.3 SP1

urlrrwriter

Type: string

Examples: c:\temp\urldiag.txt
c:\pegarules\urloutput.log

Functionality: This setting holds the name of the output log file for the *urldebug* output. The file can be stored anywhere and can be named whatever the developer wishes.

NOTE: If there is no log file name specified here, then the information is not logged, even if *urldebug* is set to record the data.

Available since: Version 5.3 SP1

useJavaIoTempDir

Type: boolean

Default: False

Functionality: This setting places the work directory underneath the value of the java property java.io.tmpdir, which is usually the same as the TEMP environment variable. The default is still the work directory supplied by the application server.

NOTE: The path to this directory may be too long; in which case select a different directory for the work dir and use the setting **explicitTempDir**. In this case, set useJavaIoTempDir to FALSE.

Available since: Version 4.2 SP2 "SmartBuild"

useNativeLibrary

Type: boolean

Default: True (version 5.1)
False (version 5.2)

Functionality: Setting this entry to FALSE will suppress loading of the pr3native.dll library. This entry was added to eliminate the error messages that occur if PegaRULES is stopped and started again in an application server (such as WebSphere); when the system is started again, the pr3native.dll library can't be loaded because the original classloader still exists (the application server doesn't clear the reference).

This setting should also be used if PegaRULES is going to be deleted and redeployed in an application server without having to restart the application server. If this library is already loaded, then the underlying files can't be deleted; the WebSphere console will report a successful new deployment, but it will have failed, and the application directory will be virtually empty.

This library is used for:

- obtaining CPU information from the operating system (if available)
- obtaining "high-resolution" elapsed time information from the operating system (if available)

- on some platforms, adds additional thread dump capabilities and heap analysis tools

Therefore, setting this entry to FALSE will disable the CPU collection in PAL.

Available since: Version 5.1

PersistRequestor subcategory

PersistRequestor

Type: string

Default: "OnTimeout"

Valid Values: "never"
"AtInteractionEnd"
"OnTimeout"

Functionality: The PersistRequestor option saves the entire context of a PRThread and/or Requestor at certain points so that upon restoration the user is presented with a "seamless" experience that doesn't require special processing by the application. For each option (except for "Never"), the context for the requestor and all of its threads will be written to the database as a System-Requestor-Context instance.

Never -- this processing is never performed.

OnTimeout – written when a requestor times out.

AtInteractionEnd – written at the end of each interactive user interaction.

NOTE: "OnTimeout" is recommended for the best tradeoff of continuity of user experience and minimal performance impact.

Available since: Version 4.2

PersistRequestor/storage

Type: string

Default: "filesystem"

Valid Values: "database"
"filesystem"

Functionality: Prior to Version 5.5, whenever a user's context was passivated, it would be stored in the database. For Version 5.5 and later, passivation to the filesystem (in the *PassivationData* directory under the *temp* directory) has been added, and is the default behavior.

This setting allows developers to set Process Commander back to the prior behavior of passivating to the database.

Available since: Version 5.5

initServices Category

The initServices section specifies whether listeners and the bootstrap CORBA service should start when Process Commander starts.

initCorba

Type: boolean

Default: False

Functionality: This entry specifies whether to start the bootstrap CORBA service on startup. If this entry is set to *true*, an ORB name server must be identified by with values for the orbInitialHost and orbInitialPort entries. The bootstrap CORBA service uses the values from orbInitialHost and orbInitialPort to register its idl file on startup.

Available since: Version 4.1

initEmail

Type: boolean

Default: True

Functionality: This entry shows whether server is set up for Email clients to connect to PegaRULES. If set to true, then all available listeners for this service (as defined in Data-Admin-Connect-EmailListener) will be enabled upon startup. Any configured and enabled listeners will be displayed in the “Listener Status” section of the PRMonitorServlet Utility.

Available since: Version 4.1

initFile

Type: boolean

Default: True

Functionality: This entry shows whether server is set up for a file-based interface to connect to PegaRULES and send information in a parsable file.

If set to true, then all available listeners for this service (as defined in Data-Admin-Connect-FileListener) will be enabled upon startup. Any configured and enabled listeners will be displayed in the "Listener Status" section of the PRMonitorServlet Utility.

Available since: Version 4.1

initJMS

Type: boolean

Default: True

Functionality: This entry specifies whether JMS listeners (Data-Admin-Connect-JMSListener) should start when Process Commander starts.

Available since: Version 4.1

initMq

Type: boolean

Default: True

Functionality: This entry shows whether server is set up for MQ clients to connect to PegaRULES. If set to true, then all available listeners for this service (as defined in Data-Admin-Connect-MQListener) will be enabled upon startup. Any configured and enabled listeners will be displayed in the "Listener Status" section of the PRMonitorServlet Utility.

Available since: Version 4.1

initJMS/pollers

Type: boolean

Default: false

Functionality: There is a subtle but important difference in the way service listeners work when Process Commander is deployed as an enterprise application and the way they work when Process Commander is deployed as a web application. This difference is especially important to the message-based listeners (MQ and JMS). To account for this difference in processing, the JMS MDB Listener was introduced: a new listener type for V5 of PRPC that is only available in an enterprise application deployment.

The use of JMS MDB Listeners are strongly recommended for all new enterprise application deployments where message-based services will be utilized, and so by default, the entries for the polling MQ and JMS listeners are not enabled (set to FALSE) in an enterprise deployment. If the polling JMS listener must be used in an enterprise deployment, this entry must be set to TRUE.

NOTE: For web application deployments, standard MQ and JMS listeners are enabled and work as they always have.

Available since: Version 5.2

initMQ/pollers

Type: boolean

Default: false

Functionality: There is a subtle but important difference in the way service listeners work when Process Commander is deployed as an enterprise application and the way they work when Process Commander is deployed as a web application. This difference is especially important to the message-based listeners (MQ and JMS). To account for this difference in processing, the JMS MDB Listener was introduced: a new listener type for V5 of PRPC that is only available in an enterprise application deployment.

The use of JMS MDB Listeners are strongly recommended for all new enterprise application deployments where message-based services will be utilized, and so by default, the entries for the polling MQ and JMS listeners are not enabled (set to FALSE) in an enterprise deployment. If the polling MQ listener must be used in an enterprise deployment, this entry must be set to TRUE.

NOTE: For web application deployments, standard MQ and JMS listeners are enabled and work as they always have.

Available since: Version 5.2

orbInitialHost

Type: string

Default: "localhost"

Functionality: This entry shows the name of the server running the Object Request Broker (ORB) transient name server process.

NOTES:

- This information must match what is specified in the ORB transient name server.
- If initCorba is set to *true*, a valid value for this entry must be provided.

Available since: Version 4.1

orbInitialPort

Type: string

Default: "7900"

Functionality: This entry specifies the port number that the ORB transient name server will use. This information must match what is specified in the ORB transient name server.

Available since: Version 4.1

license Category

This category includes settings relating to the License Compliance Tracking facility, added in Version 5.2.

enabletracking

Type: boolean

Default: true

Functionality: This setting enables or disables the run-time license/usage tracking facility.

NOTE: The exception is when using Apache Derby as the system database (i.e., when running the Process Commander Personal Edition), at which point this setting defaults to FALSE; the system does not allow this functionality to be enabled.

Available since: Version 5.2

showlicense

Type: boolean

Default: false

Functionality: This setting controls whether the license parameters are displayed in the PegaRULES console during startup, and may be used to show or confirm the license parameters that are in use. Setting this entry to *true* will display the parameters on the console.

Available since: Version 5.2

snapshotsperhour

Type: integer

Default: 1

Functionality: This setting determines how frequently the "hourly" snapshots are written. Larger values than one may be used for testing, as it

allows data to be sent to the database more quickly, allowing for quicker unit testing.

Available since: Version 5.2

writeauditdata

Type: boolean

Default: false

Functionality: This setting enables the creation of a .csv file, which is written at the end of each interaction that results in an invocation. This .csv file includes the keys of instances written to the database which caused the usage logic to decide that an invocation had occurred.

IMPORTANT: This facility is intended for use in a test environment, to allow verification that the use of the system is correctly reflected in invocation counting. Setting this entry to true will have a significant impact on performance; this setting should never be enabled in a production environment.

Available since: Version 5.2

writedatabasecsv

Type: boolean

Default: false

Functionality: This setting enables the creation of a .csv file that mirrors the hourly usage data written to the pr_hourly_usage table.

IMPORTANT: This facility is intended for use in a test environment. Setting this entry to true will have a significant impact on performance; this setting should never be enabled in a production environment.

Available since: Version 5.2

management Category

These settings allow systems to send information to the Autonomic Event Services (AES) console. AES is an independent, self-contained system that gathers, monitors, and analyzes performance and health indicators from multiple Process Commander systems. The systems being monitored send “health pulses” to the AES console – basic statistics about the health of the node, including memory, database connections, agent and requestor counts, and other data.

enabled

Type: boolean

Default: true

Functionality: This setting enables or disables the health pulse facility for this node.

Available since: Version 5.5

interval

Type: integer

Default: 120

Functionality: This setting holds the time, in seconds, between each health pulse sent from this node.

Available since: Version 5.5

notifications/appender

Type: string

Default: ALERT-AES-SOAP

Functionality: This setting holds the log4j appender name of the SOAP appender that sends alerts or pulses to the AES console.

Available since: Version 5.5

mBeanAdmin Category

The System Management Application (SMA) is defined on mBeans, and must register them with the application server in order for SMA to run in a PRPC system.

WebLogic 8.15 does not allow code to register an mBean or to get information from mBeans without presenting credentials. To register an mBean, the system must have administrative credentials.

These settings hold the administrative credentials that WebLogic 8.15 needs; SMA will not run without these settings in that appserver. For the value of these settings, consult your WebLogic administrator.

principal

Type: string

Functionality: This setting contains the administrative ID which PRPC must present to the WebLogic 8.15 application server, in order to register the SMA mBeans.

Available since: Version 5.1

Removed for: Version 5.5

credentials

Type: string

Functionality: This setting contains the credentials for the administrative ID which PRPC must present to the WebLogic 8.15 application server, in order to register the SMA mBeans.

Available since: Version 5.1

Removed for: Version 5.5

pradapter Category

This category contains settings related to logging.

logginglevel

Type: string

Default: "Info"

Valid Values: "Severe" | "Warning" | "Info" | "Config" | "Fine" | "Finer" | "Finest"

Functionality: The JVM properties file specifies the resource adapter logging level at startup. This setting may be used to overwrite the default, to state the level of tracing for output to the system log. SEVERE will provide information only about the "severe" errors; the other settings provide progressively more information, down to FINEST, which will provide the most (and should only be used for debugging). If no logging level is specified in the prconfig.xml, the Resource Adapter logger default level is inherited from the settings of the global LogManager (which are defined either through optional system properties or - if they are not set - through a lib/logging.properties file in the JRE directory.) (the latter is usually set to INFO).

Available since: Version 5.1

rulemgmt Category

This category contains settings related to managing rules and RuleSets.

stagedir

Type: string

Default: "*prpctempdir*/rulemgmt"

Example: "c:\tomcat\tempdir\rulemgmt"

Functionality: This entry is used to set the temporary staging directory for expanding or compacting the Rule Archives (used for Rules Move and other functions).

Available since: Version 5.3

runtime Category

This category contains settings related to the execution of rules.

jsp subcategory

tagpool subcategory

This subcategory contains entries which control the caching for JSP tag objects.

Example:

```
<env name="runtime/jsp/tagpool/instancecountlimit" value="100"/>
```

instanceCountLimit

Type: integer

Default: 100

Functionality: Every time a JSP tag (such as *when* or *reference* or *lookup)* is referenced in an HTML rule, an object of that tag type must be used to execute the tag. These tag objects are stored in a pool and may be reused (in order to reduce the amount of garbage created by the system).

This entry specifies how many of each type of object will be retained in the system. NOTE: If more than this number of objects are required, extra objects will be created and then discarded after use.

Available since: Version 4.2 SP5

services Category

EmailMaxSize

Type: integer

Default: -1

Functionality: This setting limits the the maximum size, in KB, for an email message. Setting this entry to "-1" allows email messages of any size.

NOTE: This setting will be overridden by the **Maximum Email Size (KB)** field on the **Process** tab of the Email Listener data instance.

Available since: Version 5.4

EmailSuppressMailAPILogging

Type: boolean

Default: false

Functionality: When developers wish to debug processes in PRPC, they turn on the debug-level logging to track issues. If the PRPC debug-level logging is enabled for email listeners, that also automatically turns on debugging in the Javamail layer, which can create a great many messages, and causes excessive output to be written to the system log.

If developers wish to see debug messages only from PRPC code (and not the underlying Java code) when they are debugging the email listener, this entry should be set to *true.*

Available since: Version 5.4

EmailSuppressUnknownContentError

Type: boolean

Default: false

Functionality: When Process Commander finds content in an email (such as embedded .jpeg files) that it can't read, by default, it will log an error message to the system log. When this entry is set to *true,* logging the error messages will be suppressed.

Available since: Version 5.4

maxRequestorChildren

Type: integer

Default: 10

Functionality: This setting limits the number of child requestors that can be created for asynchronous connection processing.

Available since: Version 5.1

soap subcategory

oldXMLPageFormat

Type: boolean

Default: False

Functionality: This setting was introduced in Version 4.2 as a result of a change that was made to the SOAP service (i.e. Rule-Service-SOAP) message format.

The SOAP message format and the generated WSDL definition of the SOAP message format were changed in such a way that some SOAP service clients developed in PRPC 4.1 would not work correctly in PRPC 4.2 – rowdata elements (used in 4.1 for all

repeating group properties) were removed. For customers who configured a SOAP service in Version 4.1 that used the XML Page parameter type, if they upgrade to 4.2 and want to continue to use these rules without reconfiguring the SOAP client, they must set this entry to TRUE. For SOAP services developed in Version 4.2 or later, this entry is not needed and should be set to false.

In order to upgrade SOAP service clients to use the new 4.2 SOAP message format without using this entry, the WSDL definition of the SOAP service must be re-published from a server category where this entry is NOT set to true. Use the WSDL to re-configure the SOAP service client. No rule changes are required.

Available since: Version 4.1

storage Category

Process Commander needs to cache data from a number of different operations to improve efficiency. In a strict J2EE application, EJBs are not allowed to directly access the file system of the hosting application server; they are used just for processing business logic and do not cache results or maintain data for future computations.

In the 5.1 release, in order to allow features requiring file access to continue to function properly, a new File Access API has been introduced, to cache the necessary data. The prconfig.xml file provides the configuration for the storage mechanisms, as well as the mapping between the resource types and those mechanisms.

Example prconfig.xml file entry:

<env name="storage/class/*resource_type:*/type" value="*mechanism*"/>

There are four different *mechanism* values, corresponding to the four storage mechanisms:

- "filesystem" – direct file system access
- "resadaptor" – file system access with resource adaptor
- "embedded" – embedded database
- "datasource" – database via JDBC DataSource

Each storage mechanism has its own particular settings, which are specified relative to the *resource_type*, so these entries will vary.

class subcategory

default

Type: string

Valid Values: "filesystem" | "resadaptor" | "embedded" | "datasource"

Functionality: A general area for the creation of files. This data type is a catchall, for any other data that might be generated. If something were incorrectly generated, it would be stored here (rather than in one of the specific types above). Ideally, there should be no data in this type, but it is provided as a safety net.

NOTES:

- If the default type is specified in the prconfig.xml file, that setting will apply to all types unless there is an additional overriding entry listing a specific type.

- If the entries in the prconfig.xml file do not match the resource_type in the file specification (typo or other issue), the system assumes the configuration to be the same as the default resource type.
- If the default configuration does not specify a storage type, it is assumed to be using the direct filesystem access off of the application's temp directory.

Available since: Version 5.1 SP1

diagnostic

Type: string

Default: "filesystem"

Functionality: Area for the storage of files generated by the JMX-based PegaRULES Management Application. Example: the Conclusion Cache report.

NOTE: This type is a special case for the 5.1 version. Since diagnostic output can be very large (one entry can be over 25MB), it is too large to store in either of the database storage mechanisms (which have a limit of 25MB per entry). Therefore, when the data is of diagnostic type, it is *always* configured to use the "filesystem" mechanism (written directly to the file system). There is no overriding of this setting.

Available since: Version 5.1 SP1

llc

Type: string

Valid Values: "filesystem" | "resadaptor" | "embedded" | "datasource"

Functionality: Area for the storage of generated output from Lookup Lists and Views.

Available since: Version 5.1

runtime

Type: string

Valid Values: "filesystem" | "resadaptor" | "embedded" | "datasource"

Functionality: Area for generated content, supporting the PegaRULES runtime code. Includes generated Java, generated classes, and generated JSPs.

The web tier does not directly execute any activities or streams; that processing is always done in the enterprise tier. Thus, the runtime data requests always go to the enterprise tier, and never touch the web tier.

Available since: Version 5.1 SP1

web

Type: string

Valid Values: "filesystem" | "resadaptor" | "embedded" | "datasource"

Functionality: Area for web-accessible files. Example: the login image.

NOTE: Simply writing a file in this resource type does not guarantee web-accessibility. The path must follow the conventions used by the static content support.

Available since: Version 5.1 SP1

timeout Category

The timeout category defines the length of time that a context can be inactive before it is passivated.

application

Type: integer

Default: 600

Functionality: This entry specifies the timeout value, in seconds, that all requestors used by services can be idle before they are passivated.

NOTE: This value does *not* include portlet services (Rule-Service-Portlet). The timeout value for Portlet services is set with the *portal* entry described below.

Available since: Version 4.1

browser

Type: integer

Default: 3600

Functionality: This entry specifies the timeout value, in seconds, that all requestors used by human users ("interactive" users) may be idle before being passivated.

Available since: Version 4.1

page

Type: integer

Default: 900

Functionality: This entry specifies the timeout value, in seconds, when pages will be passivated.

NOTE: This setting is linked to the *thread* and *browser* settings in the **timeout** category.

- If this setting is added to the prconfig file, the specified value will be used.
- If this setting is *not* specifically added to the prconfig file, but the *thread* setting is specified, the *page* setting will be half that value. Example: If the *thread* setting is specified to be 800 seconds, the *page* value will default to 400 seconds.
- If the *thread* setting is not specified, but the *browser* setting is specified, then the *thread* setting will default to half the *browser* value, and the *page* value will default to half of the *thread* setting. Example: if the *browser* setting is specified to be 1200 seconds, then the *thread* value will default to 600 seconds, and the *page* value will default to 300 seconds.

Available since: Version 5.5

portal

Type: integer

Default: 3000

Functionality: This entry specifies the timeout value, in seconds, that all requestors used by portlet services (Rule-Service-Portlet) may be idle before being passivated. NOTE: This entry should be set to the same value as the "browser" setting.

Available since: Version 4.1

thread

Type: integer

Default: 1800

Functionality: This entry specifies the timeout value, in seconds, that threads within a requestor may be idle before being passivated.

NOTE: This setting is linked to the *browser* settings in the **timeout** category.

- If this setting is added to the prconfig file, the specified value will be used.
- If this setting is *not* specifically added to the prconfig file, but the *browser* setting is specified, the *thread* setting will be half that value. Example: If the *browser* setting is specified to be 800 seconds, the *thread* value will default to 400 seconds.

Available since: Version 5.5

tracer Category

The Tracer category contains entries which relate to the tracer functionality.

timeout

Type: integer

Default: 3600

Functionality: This entry specifies the amount of time (in seconds) that a tracer session may remain idle (without interaction from the user) before it is closed.

Available since: Version 5.3

queue subcategory

The tracer queue is where the tracer stores all the information being generated. These settings relate to that queue.

type

Type: string

Valid Values: "memory" | "file"

Functionality: In Version 5.3, changes were made to the tracer to improve usability and address the "Buffer Overflow" warning. The buffer overflow occurs while tracing large activities and flows, and causes the user to miss an unknown number of trace events. These overflows happen because the in-memory queue in which tracer stores its events can only hold so many events before risking out-of-memory exceptions. To solve this issue, developers can now choose to store tracer information either in memory or in a file.

This setting determines where the tracer information will be stored.

Available since: Version 5.3

file/limit

Type: integer

Valid Values: any value greater than or equal to zero

Default: 50,000

Functionality: This setting specifies the number of entries allowed in one tracer file; it was created to control “runaway” or “infinite loop” tracer processes. Once the tracer generates the specified number of entries, the tracer is stopped.

If set to “0” (zero), there is no limit on the number of entries in a tracer file. Any value greater than zero will set that number as the limit of entries.

Available since: Version 5.3

Usage Category

The Usage category contains entries which relate to the System Usage History report (available through the Monitor Servlet).

RetentionPeriod

Type: integer

Default: 30

Functionality: This setting shows the number of days that the performance usage information entries will be kept in the database before being purged.

Available since: Version 5.2

HistoryReport subcategory

RegularUserThreshold

Type: string

Default: "4"

Functionality: In the System Usage report, the Named Users are divided into Regular and Occasional users, based on their usage. This entry holds the threshold, measured in hours, which determines whether the user is a Regular or Occasional user.

Available since: Version 4.2 SP2 "SmartBuild"

CustomQuery subcategory

With these entries, developers may change the System Usage History report to use their own custom-written SQL.

JDBCString category

Due to the fact that the SQL code must be different for each relational database vendor (i.e., Oracle, MSSQL, DB2), the "vendor string for the JDBC driver" must be used as a subcategory for these entries, to identify the appropriate Rule-Connect-SQL instance.

Example:

```
<env name="usage/HistoryReport/CustomQuery/Oracle/BPMInvocationCount"
value=" AcmeCoBPMInvocationCount "/>
<env name="usage/HistoryReport/CustomQuery/Oracle/NamedUserCount"
value=" AcmeCoNamedUserCount "/>
```

BPMInvocationCount

Type: string

Example: "AcmeCoBPMInvocationCount"

Functionality: This entry allows the developer to customize the SQL used in the System Usage History report for the BPM Invocations.

Available since: Version 4.2 SP2 "SmartBuild"

NamedUserCount

Type: string

Example: "AcmeCoNamedUserCount"

Functionality: This entry allows the developer to customize the SQL used in the System Usage History report for the Regular Named Users and the Occasional Named Users.

Available since: Version 4.2 SP2 "SmartBuild"

PeakUserCount

Type: string

Example: "AcmeCoPeakUserCount"

Functionality: This entry allows the developer to customize the SQL used in the System Usage History report for the Users in Peak Hour.

Available since: Version 4.2 SP2 "SmartBuild"

RuleInvocationCount

Type: string

Example: "AcmeCoRuleInvocationCount"

Functionality: This entry allows the developer to customize the SQL used in the System Usage History report for the Rule Invocations.

Available since: Version 4.2 SP2 "SmartBuild"

Rules subcategory

These settings collect data for performance usage tracking.

committhreshold

Type: integer

Default: 100

Functionality: This setting configures the number of snapshot entries which are collected before a commit is performed. The larger this number, the more system resources are required to do the commit. Levels higher than the default may be used on more robust (fast) systems to improve snapshot performance.

Available since: Version 5.2

snapshotonshutdown

Type: boolean

Default: true

Functionality: The new Rules Usage reports give a historical view of what rules are being used when processes run, to allow customers to see what rules are being used, and what should be deleted. This setting determines whether a snapshot of usage is taken when the system shuts down. NOTE: When enabled, this snapshot will only occur when the system shuts down gracefully/correctly.

Available since: Version 5.2

Chapter 3: Dynamic System Settings

"Classic" Settings

email/blockResponse

Owning RuleSet: Pega-IntSvcs

Type: boolean

Default: false

Created in system by: added by application developer if setting needs to be changed

Functionality: This setting is used by a file listener. Some customers do not wish to send their messages through SMTP, and have turned off that functionality. However, the system was still trying to make this outbound connection, and then shutting down the listener when the connection failed.

This setting should be enabled when the listener is *not* supposed to send responses.

Available since: Version 5.5

email/blockResponseContent

Owning RuleSet: Pega-IntSvcs

Type: boolean

Default: false

Created in system by: added by application developer if setting needs to be changed

Functionality: This setting is used by a file listener. When there is an error in the process, the system will send the text of that error to an administrator or display it in the console log. The contents of the

message were displayed in this error, even though those contents (meant to be sent to one customer, for example) may be proprietary.

This setting allows the text of the original message (involved when the error occurred) to be suppressed.

Available since: Version 5.5 SP2

fua/minimumAgeToPurgeFromDatabase

Owning RuleSet: Pega-RulesEngine

Type: integer

Default: 7

Created in system by: added by application developer if setting needs to be changed

Functionality: This setting holds the minimum amount of time (in days) for an FUA cache entry to be purged.

Available since: Version 6.1

general/maxRuleBrowse

Owning RuleSet: Pega-IntSvcs

Type: integer

Default: 50

Created in system by: added by application developer if setting needs to be changed

Functionality: This setting will override the default limit on the total number of listeners that can be found at start up.

Available since: Version 5.4 SP2

mq/gmo

Owning RuleSet: Pega-IntSvcs

Type: integer

Default: -1

Created in system by: added by application developer if setting needs to be changed

Functionality: This setting allows the user to add additional Websphere MQ “get message options” (GMO) for MQ Services. There is a large list of GMO available in the MQ documentation – the user would set the value of our system setting to the sum of the individual GMO options desired.

Available since: Version 5.4

mq/pmo

Owning RuleSet: Pega-IntSvcs

Type: integer

Default: -1

Created in system by: added by application developer if setting needs to be changed

Functionality: Allows the user to add additional Websphere MQ “put message options” (PMO) for MQ Services. There is a large list of PMO available in the MQ documentation – the user would set the value of our system setting to the sum of the individual PMO options desired.

Available since: Version 5.4

mq/oo

Owning RuleSet: Pega-IntSvcs

Type: integer

Default: -1

Created in system by: added by application developer if setting needs to be changed

Functionality: Allows the user to add additional Websphere MQ “open options” (OO) for the request queue and response queue in MQ Services. There is a large list of OO available in the MQ documentation – the user would set the value of our system setting to the sum of the individual OO options desired.

Available since: Version 5.4

ProComHelpURI

Owning RuleSet: Pega-ProCom

Type: string

Default: *blank*

Example: http://*myserver*/prhelp

Created in system by: the installation process

Functionality: This setting holds the URL of the server where the on-line help files have been installed.

Available since: Version 5.1

PublicLinkURL

Owning RuleSet: Pega-ProCom

Type: string

Default: *blank*

Example: **http://*myserver*/*contextroot*/PRServlet**

Created in system by: the installation process

Functionality: This setting holds the URL to the system. It is used in correspondence – whenever an email is sent to someone from the Process Commander system, it includes a link back to the system for a reply.

In prior versions, the link was directly to the server from which the message came. If there were a multi-node setup, this link might point directly at one of the sub-nodes, not to the overall system URL. This setting allows the developer to specify a URL that all correspondence should use to reply to messages from that system.

Available since: Version 5.2

SearchSoapURI

Owning RuleSet: Pega-RULES

Type: string

Default: *blank*

Example: http://*myserver*/*contextroot*/PRSOAPServlet

Created in system by: *shipped* – value set as choice on First Steps menu

Functionality: This setting holds the URL of the one node which contains the Lucene index files for the multi-node Process Commander installation. (See *indexing,* below.) In the multi-node setup, all the nodes in the system which do *not* have the index files must connect to the indexed node via a SOAP connection.

Available since: Version 5.1

soap/SR-17635

Owning RuleSet: Pega-IntSvcs

Type: boolean

Default: false

Created in system by: added by application developer if setting needs to be changed

Functionality: This feature affects WSDL generation for SOAP services. When turned on, type names will match class names in PRPC, and values from table edits will be reflected in the WSDL. This setting should not be needed in 5.5 as the standard WSDL generation was improved and this switch should not be necessary.

Available since: Version 5.4. SP1

Deprecated in: Version 5.5

SOAPServicePreserveWhiteSpaces

Owning RuleSet: Pega-IntSvcs

Type: boolean

Default: false

Created in system by: added by application developer if setting needs to be changed

Functionality: This feature affects the processing of SOAP requests in SOAP services. When turned on, trailing whitespace in the xml element data is not trimmed before mapping to the clipboard.

Available since: Version 5.4

SystemManagementURI

Owning RuleSet: Pega-ProCom

Type: string

Default: *blank*

Example: http://*myserver*/prsysmgmt

Created in system by: the installation process

Functionality: This setting holds the URL of the server where the System Management Application (SMA) has been installed.

Available since: Version 5.1

WorkHistoryVersion

Owning RuleSet: Pega-ProCom

Type: string

Default: 4.2

Valid Values: 4.2
5.1

Created in system by: the installation process

Functionality: The Work History Upgrade Utility is a feature provided in Process Commander Version 5.1 to allow customers to take advantage of performance improvements related to retrieving history data for display.

Customers who have Version 4.2 systems and who have upgraded to Version 5.1 currently display history data in the slower, “Version 4.2” implementation. If they desire, they can switch their system to the new, faster “Version 5.1” implementation, by using the History Upgrade Utility.

Available since: Version 5.1

xml/maxNodeCount

Owning RuleSet: Pega-IntSvcs

Type: integer

Default: 100

Created in system by: added by application developer if setting needs to be changed

Functionality: This setting is used by the XSD Import Accelerator and the SOAP Connector Accelerator. When XML Parse or XML Stream rules are generated, if the number of attributes or child elements of the XML definition exceed the value of this entry, the rule is broken into linked components (to avoid having too large a rule).

Available since: Version 5.5

compiler

The compiler category enables you to add Java classes to the Process Commander compile time classpath.

defaultClasses

Owning RuleSet: Pega-RULES

Type: string

Default: javax.jms.TopicSession;javax.portlet.GenericPortlet;org.apache.commons.httpclient.Credentials;com.ibm.mq.MQC

Example: org.apache.SOAP.AttributeHandler

Created in system by: the installation process

Functionality: This setting holds the list of compile time classes to be included in the application server classpath at runtime. The list should be separated by semicolons, and the name of the classes should be separated by either a period (“.”) or a forward slash (“/”). The end of the name should *not* end in “.class,” but just the class name should be referenced (“com.pega.pegarules.engine.getCompilerInfo”). Quotes are not needed around the list.

For any classes referenced in the System Settings, the .jar file containing those classes should be stored in the /WEB-INF/lib directory (for a WAR file) or the /APP-INF/lib directory (for an EAR file).

After classes are added to this setting, they must be read by the system. This can be done at startup (stopping and starting the system). If the developer doesn’t wish to stop and start the application, then they may use the System Management Application to refresh the classes. Under the **Advanced** section, on the **ClassLoader Management** page, click the **Refresh External Jars** button.

Available since: Version 5.1

defaultPaths

Owning RuleSet: Pega-RULES

Type: string

Default: *blank*

Example: c:/websphere/appserver/jarfiles/SOAP.jar

Created in system by: the installation process

Functionality: This setting holds the list of compile time .jar files (with their full pathname) to be included in the application server classpath at runtime. The list should be separated by semicolons.

Since this listing includes the full path to the .jar file, these files may be put anywhere in the system. When the classes are loaded, the .jar files listed in the *defaultPaths* entry of the prconfig.xml file will be loaded first, and then the .jar files specified in this setting.

IMPORTANT NOTE: For multi-node systems, since the Dynamic System Settings are read by the entire system (not just one node), the .jar file must be put in the same place on *each* node.

After classes are added to this setting, they must be read by the system. This can be done at startup (stopping and starting the system). If the developer doesn't want to stop and start the application, then they may use the System Management Application to refresh the classes. Under the **Advanced** section, on the **ClassLoader Management** page, click the **Refresh External Jars** button.

Available since: Version 5.1

indexing

In order to make finding specific rules easier in the PegaRULES Process Commander system, a full-text search has been implemented, based on the Apache Lucene open source indexing software. The *indexing* subcategory of settings control the process of indexing the database. Separate index files are used for rules, data instances, and work objects.

In Version 5.1, enhancements were made to the Lucene indexing functionality for multi-node systems. The new functionality allowed the developer to designate *one* node of a Process Commander system to store the index files; all other systems would point to that node, and the entire system would use one index. In this case, all the prconfig.xml settings would be *ignored*; instead, Dynamic System Settings would be used (so all nodes would have the same server information).

enabled

Owning RuleSet: Pega-RULES

Type: boolean

Default: true

Created in system by: *shipped*

Functionality: This entry allows globally enabling or disabling rule indexing on the Process Commander system. If the entry is set to True, then changes to rules, work, or data may be indexed via the Lucene functionality (depending upon their individual class settings - ruleEnabled, workEnabled, dataEnabled).

This index will be stored either in the explicitIndexDir (if specified), or written to a directory under the context root: /prweb/PegaRULESIndex.

Available since: Version 5.1

attachmentenabled

Owning RuleSet: Pega-RULES

Type: boolean

Default: false

Created in system by: *shipped*

Functionality: This entry allows globally enabling or disabling indexing of attachments to work objects in the Process Commander system. If the entry is set to True, then all work item attachments may be indexed via the Lucene functionality.

Available since: Version 6.1

dataEnabled

Owning RuleSet: Pega-RULES

Type: boolean

Default: false

Created in system by: *shipped*

Functionality: This entry enables or disables indexing for Data- classes. NOTE: This entry is only relevant if the "enabled" entry is set to TRUE.

Available since: Version 5.1

explicitindexdir

Owning RuleSet: Pega-RULES

Type: string

Example: “C:\index”

Created in system by: *shipped* – value set as choice on First Steps menu

Functionality: This setting can be used to force PegaRULES to write its index files to a certain location. When not specified, this value defaults to a sub-directory of the PegaRULES temp directory called PegaRULESIndex. The value specified MUST be an absolute file path, and there must be at least 300MB of free space available in the directory. (The index file will grow as changes and additions are made to rules.)

NOTE: Index directories should not be shared across distinct PegaRULES installations.

Available since: Version 5.1

hostid

Owning RuleSet: Pega-RULES

Type: string

Default: empty

Example: "52891d2344e2e25b1827bc7692825b8f"

Created in system by: *shipped* - set by system processing

Functionality: This entry holds the node ID of the host machine where the single index for the Process Commander system will be stored.

IMPORTANT: This setting is created and maintained by the system and *should never be changed.*

Available since: Version 5.1

hostname

Owning RuleSet: Pega-RULES

Type: string

Default: empty

Example: "myserver"

Created in system by: *shipped* – value set as choice on First Steps menu

Functionality: This entry holds the name of the host ID server where the single index for the Process Commander system will be stored.

NOTE: If the customer wishes to change the "host" node after the install process is complete, and have the index stored on a different node, this setting must be changed. *NEVER change the hostid setting.* After the change is made in this setting, the server must be stopped and restarted, and the indexes must be rebuilt on the new host. (The developer could also move the index files from the old node to the new node, but then must run the System-Status-Nodes.*UpdateIndexBuildStatus* activity.)

Available since: Version 5.1

ruleEnabled

Owning RuleSet: Pega-RULES

Type: boolean

Default: true

Created in system by: *shipped*

Functionality: This entry enables or disables indexing for Rule- classes. NOTE: This entry is only relevant if the "enabled" entry is set to TRUE.

Available since: Version 5.1

searcherror

Owning RuleSet: Pega-RULES

Type: boolean

Default: false

Created in system by: *shipped*

Functionality: This setting determines whether an error message will be displayed if indexing is not currently working. If this entry is set to TRUE, and there is an indexing problem, then when any search is done, a message will be displayed at the top of the search results stating that "indexing is not complete." (Searching will continue to work – it will just search the things which had already been indexed. This error indicates that since indexing is not complete, some items may not be indexed yet.) This feature is on a five-minute cycle to check whether indexing has been reenabled.

NOTE: The FALSE setting for this entry is just for the initial configuration. If there is an error, this setting will dynamically change to TRUE. That is fine – do not change it back to FALSE.

Available since: Version 6.1

updatetime

Owning RuleSet: Pega-RULES

Type: string

Default: empty

Example: "20061030T163003.860 GMT"

Created in system by: shipped

Functionality: This entry holds the timestamp of the last time the index was updated (which was the last time the System Pulse ran and found changes to the rules).

IMPORTANT: This setting is created and maintained by the system and *should never be changed.*

Available since: Version 5.1

workEnabled

Owning RuleSet: Pega-RULES

Type: boolean

Default: false

Created in system by: *shipped*

Functionality: This entry enables or disables indexing for Work- classes. NOTE: This entry is only relevant if the "enabled" entry is set to TRUE.

Available since: Version 5.1

"Data-defined prconfig" Settings (shipped)

prconfig/agent/enable/default

Owning RuleSet: Pega-Engine

Type: boolean

Default: true

Functionality: Setting this entry to *true* enables application agents to run on this category.

Setting this entry to *false* will disable all agents.

Available since: Version 6.2

prconfig/database/baseTable/name/default

Owning RuleSet: Pega-Engine

Type: string

Default: "pr4_base"

Functionality: This entry holds the name for base table for system startup information.

Available since: Version 6.2

prconfig/identification/systemName/default

Owning RuleSet: Pega-Engine

Type: string

Default: "pega"

Functionality: Name of instance in Data-Admin-System for this server. Also affects System Pulse; defines the scope of response to messages that are sent out via the pulse. Systems can send messages that

will only affect their own system (possibly across multiple machines) or all systems that share a database across all machines.

Available since: Version 6.2

prconfig/initServices/initEmail/default

Owning RuleSet: Pega-Engine

Type: boolean

Default: True

Functionality: This entry shows whether server is set up for Email clients to connect to PegaRULES. If set to true, then all available listeners for this service (as defined in Data-Admin-Connect-EmailListener) will be enabled upon startup. Any configured and enabled listeners will be displayed in the "Listener Status" section of the PRMonitorServlet Utility.

Available since: Version 6.2

prconfig/initServices/initFile/default

Owning RuleSet: Pega-Engine

Type: boolean

Default: True

Functionality: This entry shows whether server is set up for a file-based interface to connect to PegaRULES and send information in a parsable file. If set to true, then all available listeners for this service (as defined in Data-Admin-Connect-FileListener) will be enabled upon startup. Any configured and enabled listeners will be displayed in the "Listener Status" section of the PRMonitorServlet Utility.

Available since: Version 6.2

prconfig/initServices/initJMS/default

Owning RuleSet: Pega-Engine

Type: boolean

Default: True

Functionality: This entry specifies whether JMS listeners (Data-Admin-Connect-JMSListener) should start when Process Commander starts.

Available since: Version 6.2

prconfig/initServices/initMQ/default

Owning RuleSet: Pega-Engine

Type: boolean

Default: True

Functionality: This entry shows whether server is set up for MQ clients to connect to PegaRULES. If set to true, then all available listeners for this service (as defined in Data-Admin-Connect-MQListener) will be enabled upon startup. Any configured and enabled listeners will be displayed in the "Listener Status" section of the PRMonitorServlet Utility.

Available since: Version 6.2

prconfig/initialization/displayExceptionTraceback/default

Owning RuleSet: Pega-Engine

Type: boolean

Default: false

Functionality: When an error occurs in Process Commander, the exception screen is displayed. Clicking on the **Show Exception Details** button will show data which includes stack trace messages, which can contain system details that may be useful to users trying to circumvent system security.

Even if the button is not clicked, the stack trace information may be displayed by right-clicking on the browser screen and using the **View Source** option.

This setting controls the display of the stack trace information. When set to *false,* this entry will remove the **Show Exception Details** button from the error screen, and also remove all the stack trace data from the **View Source** screen.

Available since: Version 6.2

prconfig/initialization/explicitTempDir/default

Owning RuleSet: Pega-Engine

Type: string

Default: "${pega.tmpdir}"

Functionality: This setting specifies the full path which should be used as the work directory. This allows the developer to use a shorter path to the temp (work) directory, in order to avoid exceeding various platform path name limits. The default continues to be the work directory supplied by the application server.

Important Notes:

- Specifying the *explicittempdir* setting in the prconfig.xml file but leaving the value blank will cause the system to create the temp (work) directory in the default location supplied by the application server. *A value must be specified here in order for the temp path to be shortened.*
- If this entry is used, then the useJavaIoTempDir entry must be set to FALSE.
- The value specified by this entry will be ignored unless the directory already exists. Make sure that the user has created the directory specified prior to starting PegaRULES. If the directory does NOT exist, then the temp directory provided by the application server will be used and no warning or error will be logged.
- It is possible to use Java property substitutions to specify this setting as a JVM system variable (example: ${*setting*}). In WebSphere, the shipped WAS policy file expects a Java

property named **pega.tmpdir** to point to the explicit temp directory location, so using this property name in conjunction with the substitution (e.g., ${pega.tmpdir}) would keep the reference consistent.

Available since: Version 6.2

prconfig/initialization/persistRequestor/default

Owning RuleSet: Pega-Engine

Type: string

Default: "OnTimeout"

Valid Values: "never"
"AtInteractionEnd"
"OnTimeout"

Functionality: The PersistRequestor option saves the entire context of a PRThread and/or Requestor at certain points so that upon restoration the user is presented with a "seamless" experience that doesn't require special processing by the application. For each option (except for "Never"), the context for the requestor and all of its threads will be written to the database as a System-Requestor-Context instance.

Never -- this processing is never performed.

OnTimeout – written when a requestor times out.

AtInteractionEnd – written at the end of each interactive user interaction.

NOTE: "OnTimeout" is recommended for the best tradeoff of continuity of user experience and minimal performance impact.

Available since: Version 6.2

prconfig/pradapter/loggingLevel/default

Owning RuleSet: Pega-Engine

Default: "Info"

Valid Values: "Severe" | "Warning" | "Info" | "Config" | "Fine" | "Finer" | "Finest"

Functionality: The JVM properties file specifies the resource adapter logging level at startup. This setting may be used to overwrite the default, to state the level of tracing for output to the system log. SEVERE will provide information only about the "severe" errors; the other settings provide progressively more information, down to FINEST, which will provide the most (and should only be used for debugging). If no logging level is specified in the prconfig.xml, the Resource Adapter logger default level is inherited from the settings of the global LogManager (which are defined either through optional system properties or - if they are not set - through a lib/logging.properties file in the JRE directory.) (the latter is usually set to INFO).

Available since: Version 6.2

"Data-defined prbootstrap" Settings (shipped)

properties/com.pega.pegarules.bootstrap.codeset.version.Customer/default

Owning RuleSet: Pega-Engine

Type: string

Default: 06-01-01

Functionality: This entry holds the version of the Customer CodeSet.

Available since: Version 6.2

properties/com.pega.pegarules.bootstrap.codeset.version.Pega-EngineCode/default

Owning RuleSet: Pega-Engine

Type: string

Default: 06-02-01

Functionality: This entry holds the version of the engine code CodeSet.

Available since: Version 6.2

Chapter 4: System Settings

The **System Settings Per Production Level** shows the values at each production level. There are five system levels in Process Commander:

Level	Description
1	experimental
2	development
3	test
4	pre-production
5	production

BPCleanUpInterval

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	5
2	5
3	5
4	5
5	5

Functionality: The bulk processing agent knows that a job is complete when the total assignments processed equals the total assignments requested. However, if another agent (like service level) or an end

user updates or deletes an assignment before bulk processing can process it, then the total assignments processed will never add up to what was requested. This is known as a stale or stalled bulk processing job.

When a bulk processing job is stalled, this is the interval, in minutes, which the system will wait before finishing up the job and declaring all unfinished assignments skipped.

Available since: Version 5.3

BPUnitsToProcess

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	50
2	50
3	50
4	50
5	50

Functionality: Controls the number of Assigments to process in an agent interval. These should be numeric values.

Available since: Version 5.3

BPUnitsToProcessForeground

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	1
2	1
3	1
4	1
5	1

Functionality: The Bulk Processing feature will to process at least one assignment in the foreground (not in the background), to be sure that the bulk processing of these assignments was correctly set up by the user.

Available since: Version 5.3

BulkProcessTransferFlowAction

Owning RuleSet: Pega-ProCom

Type: string

Defaults:

Level	Default value
1	transfer
2	transfer
3	transfer
4	transfer
5	transfer

Functionality: By default, the Transfer Flow Action choice on the WorkManager portal uses the *transfer* Flow Action. This setting allows a developer to specify a different flow action to use for this process (in case the customer has called their transfer process by a different name).

Available since: Version 5.4

BulkProcessingID

Owning RuleSet: Pega-ProCom

Type: string

Defaults:

Level	Default value
1	pyBulkProcessing
2	pyBulkProcessing
3	pyBulkProcessing
4	pyBulkProcessing
5	pyBulkProcessing

Functionality: pxAssignedOperatorID is set to this ID on selected assignments when bulk processing is submitted.

NOTE: No other user or basket in the system should have this ID.

Available since: Version 5.3

FilterHistory

Owning RuleSet: Pega-ProCom

Type: string

Defaults:

Level	Default value
1	WRITE_NONE
2	WRITE_SOME
3	WRITE_ALL
4	WRITE_ALL
5	WRITE_ALL

Functionality: This setting allows a process architect to turn on, turn off, or select a custom setting for ProCom work history writing. Note that changes to this rule DO NOT TAKE EFFECT IMMEDIATELY. You must run the activity History-Work-.resetFilterHistory or restart the server.

Possible values for FilterHistory are:

- WRITE_ALL - all work history is written (backwards compatible mode)
- WRITE_SOME - checks the FilterHistory decision tree to determine case-by-case basis
- WRITE_NONE - all ProCom-written history is skipped. History added by customer rules(through the use of History-Add or the Audit Note field) is not affected.

Available since: Version 5.4

GetNextWork_MoveAssignmentToWorklist

Owning RuleSet: Pega-ProCom

Type: boolean

Defaults:

Level	Default value
1	true
2	true
3	true
4	true
5	true

Functionality: If set to false, GetNextWork will keep workbasket assignments in the workbasket and not move them to the user's list.

Available since: Version 5.4

GetNextWork_WorkBasketUrgencyThreshold

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	0
2	0
3	0
4	0
5	0

Functionality: GetNextWork looks in the workbaskets in the order listed on the operator id form. If an urgency threshold above 0 is specified, it will move to the next workbasket when no assigments with that urgency or higher are found. Only after all the workbaskets listed are drained of assignments with urgency above the threshold will it come back to the first workbasket and get the less important assignments.

A value of zero indicates that this functionality is not being used. (This is the default.)

Available since: Version 5.1

Deprecated in: Version 5.5 (set in the Operator ID in that version)

MaxFlowEnteredCount

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	500
2	500
3	500
4	500
5	500

Functionality: This setting guards against infinite loops in a flow. A property on the pxFlow page called pyFlowCalledCount increments every time a flow is resumed, and is reset to 1 at every stopping point (be it assignment or subflow).

When pyFlowCalledCount reaches the number held in this setting, the flow stops retrying and goes into a problem flow.

Available since: Version 5.2

MaxFlowLoopCount

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	500
2	500
3	500
4	500
5	500

Functionality: This setting guards against infinite loops in a flow. A typical infinite loop is where shape A calls shape B which calls shape A again, without an intervening stopping point such as an assignment.

When the number of calls from one shape to another reaches the value held in this setting, the flow stops retrying and goes into a problem flow.

NOTE: Only set this value higher than 500 if you have an iteration that requires it.

Available since: Version 5.2

PegaPDNQueryURI

Owning RuleSet: Pega-WB

Type: string

Defaults:

Level	Default value
1	http://pdn.pega.com/Query.asp?Release=V62&ID=
2	http://pdn.pega.com/Query.asp?Release=V62&ID=
3	http://pdn.pega.com/Query.asp?Release=V62&ID=
4	http://pdn.pega.com/Query.asp?Release=V62&ID=
5	http://pdn.pega.com/Query.asp?Release=V62&ID=

Functionality: Contains the URI to query the Pega PDN for certain topics linked in Process Commander. This setting provides product version information to the PDN, so the correct article for the customer's version is displayed.

Available since: Version 5.4

ThrowExceptionOnLockError

Owning RuleSet: Pega-ProCom

Type: boolean

Defaults:

Level	Default value
1	true
2	
3	
4	
5	

Functionality: This setting is used by the activity Assign-Corr>SendCorr to throw exception in the case of a database lock. If the value of this setting is false, then the SendCorr activity logs the problem in an error log instead of throwing an exception.

Available since: Version 6.1

pyUseGoalTimeForAvailabilityCalculation

Owning RuleSet: Pega-ProCom

Type: boolean

Defaults:

Level	Default value
1	true
2	true
3	true
4	true
5	true

Functionality: If true, the assignment's goal time is compared to the assignment's start time and effort estimate to determine operator availability.

For instance, if the start time is Feb 15th, the goal time Feb 22nd, the effort estimate 3 business days, then if the operator has 3 vacation days scheduled in that time, he won't be considered available for the assignment (5 business days between dates, only 2 working days open for 3 days of work). The substitute operator's availability would then be checked.

Available since: Version 6.1

SLA settings

By default, the SLA agent runs every 30 seconds to process assignments. You should tune the SLA processing to be as efficient as possible, by determining:

- how many assignments should be *processed* in one interval
- how many assignments should be *retrieved* for one interval's processing

These settings will vary depending upon:

- how many nodes the Process Commander system is running on (single or multiple – and is "multiple" 2 nodes, or 200?)
- how many work objects with SLAs are created each day

For a full description of the tuning process, please reference the KB article *System Settings for the Pega-ProCom SLA Agent*.

NOTE: Beginning in Version 5.5, these settings are not used (replaced by the Queue Manager).

SLALockRetryInterval

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	90,000
2	90,000
3	90,000
4	90,000
5	90,000

Functionality: When the queue manager reaches a work object which is locked, how much later should it retry it? This setting holds the time value, in seconds, for when the next retry occurs (default is 15 minutes).

Available since: Version 5.5

SLARefreshListEachIteration

Owning RuleSet: Pega-ProCom

Type: boolean

Defaults:

Level	Default value
1	false
2	false
3	false
4	false
5	false

Functionality: If this setting is *true*, then each time an SLA agent successfully processes an assignment, it refreshes the retrieved item list (retrieves a new list of assignments).

NOTE: On a single node system, this should be set as false (and SLAUnitsToRetrieve and SLAUnitsToProcess should be large). On a multi-node system, this setting should be true and SLAUnitsToRetrieve should be small.

Available since: Version 5.1

Deprecated in : Version 5.5

NOTE: Beginning in Version 5.5, this setting is not used (replaced by the Queue Manager).

SLAUnitsToProcess

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	1
2	1
3	1
4	1
5	1

Functionality: This setting controls the number of Assigments the SLA agent will process during its interval.

Available since: Version 5.1

Deprecated in : Version 5.5

NOTE: Beginning in Version 5.5, this setting is not used (replaced by the Queue Manager).

SLAUnitsToRetrieve

Owning RuleSet: Pega-ProCom

Type: integer

Defaults:

Level	Default value
1	5
2	5
3	5
4	5
5	5

Functionality: This setting shows the number of assignments the SLA agent will retrieve for processing.

NOTE: There are three types of SLA events for assignments:

- goal
- deadline
- late

This setting reflects the number of events handled *for each type*. Therefore, if this entry is set to "5," the SLA agent could potentially process up to 15 assignments – 5 for *each* type.

Available since: Version 5.1

Deprecated in : Version 5.5

NOTE: Beginning in Version 5.5, this setting is not used (replaced by the Queue Manager).

Appendix A: Logic Flow for Uploading Settings into Database