PEGA 7 PLATFORM

Pega Platform

Update Guide

7.3.1

Pegasystems Inc.
One Rogers Street
Cambridge, MA02142-1209
USA
Phone: 617-374-9600
Fax: (617) 374-9620
www.pega.com

DOCUMENT: Pega Platform Update Guide
SOFTWARE VERSION: 7.3.1
PUBLISHED: Monday, April 16, 2018

CONTENTS

Plan your deployment

The Pega Platform supports several configuration options that can affect the choices that you make during the deployment. Before beginning, read this section thoroughly.

- Do not change your environment while you are deploying the Pega Platform. For example, if you are making changes to your application server, or database server, do so before you deploy the Pega Platform.
- Choose a configuration type: single-schema or split-schema configuration. Pegasystems recommends a split-schema configuration. See Schema configuration options.
- Choose whether to use the standard product edition or the multitenancy edition. The multitenancy edition has different requirements, different run-time behaviors, and different administrative procedures from the standard edition. Before you select the multitenancy edition, review the *Multitenancy Administration Guide* on the PDN.

  Upgrading and updating from one edition to another is not supported. If you install one edition and later decide to use a different edition, you must drop and re-create the database or create a new database. The schema DDLs for the two editions are not compatible.
- Choose a clustering topology: Hazelcast or Apache Ignite; standard embedded mode or client-server mode. Embedded Hazelcast is the default clustering topology. If you want to use Apache Ignite clustering topology, you need to enable Apache Ignite cluster protocol in **prconfig.xml** file. You can use Apache Ignite embedded mode only for small clusters. If you want to use Apache Ignite client-server mode, you need to additionally force the Pega Platform node to start in client mode and open ports for Apache Ignite. See Apache Ignite client-server clustering topology.
- Verify the Business Intelligence Exchange (BIX) and Pega Platform product versions. Release versions of the Business Intelligence Exchange (BIX) are synchronized with release versions of the Pega Platform. BIX is included in the full distribution image, but has a separate installer. Verify that the version of BIX is the same as the version of the Pega Platform. For information about installing BIX, see the Pega Platform *BIX User Guide*.
- Choose a deployment type: UI tool or command line. See Deployment methods.
- Choose whether to use Kerberos functionality. Kerberos is a computer network authentication protocol that allows nodes communicating over a non-secure network to prove their identity to

one another in a secure manner. If you enable Kerberos authentication, you must use the command line to deploy the Pega Platform.

- Consult with your database administrator to determine whether to have the deployment process make changes directly to the database. You can either have Pega Platform apply changes directly to your database, or generate DDL files of changes for your database administrator to apply. For information about manually generating and applying DDL, see Appendix C — Optional: Generating and applying DDL.
- Choose whether to cluster the Pega Platform nodes. The Pega Platform supports clustered nodes without special configuration, but you will make different choices about ports, indexes, and clock synchronization depending on your node configuration.
- Determine the best configuration for your database. Involve your database administrator in these decisions. For more information, see PDN > Support.
  - For split-schema configurations, choose whether you will maintain separate tablespaces for the data schema and rules schema. This decision depends on your database configuration.
  - Conduct a site-dependent analysis of how the Pega Platform and any Industry applications that will be used to determine the size of your database tablespace.
- Choose either dual-user or single-user configuration. In a dual-user configuration, an Admin user is granted full privileges, and a Base user is granted a smaller subset of privileges. In the single-user configuration, a single Base user is granted full privileges.

  For more information about user configuration, see your installation guide.
- If you are using the PostGIS extension on a PostgreSQL database, ensure that it has not been applied to the Rules or Data schemas because it will cause the deployment to fail.

Split-schema and single-schema configurations

There are two configuration types: single schema and split-schema. Pegasystems recommends split-schema configurations, particularly in critical development environments such as quality assurance, staging, and production.

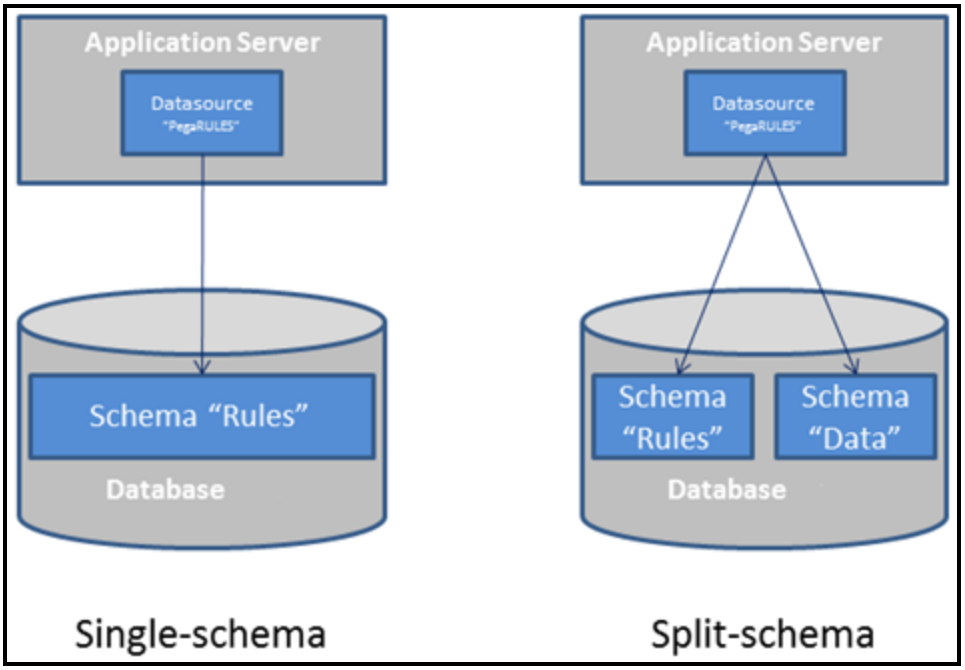
- Single-schema configuration — One schema contains all rules and data objects.
- Split-schema configuration — The rules and data objects reside on separate schemas:

- A Rules schema contains rules tables and associated data and objects.
- A Data schema contains transaction data, including work objects.

With a split-schema configuration, you can upgrade one environment, and then migrate the upgraded objects to other environments.

In a split-schema configuration, the Pega Platform uses the Java Naming and Directory Interface (JNDI) standard to identify and access the appropriate schema. One of the benefits of using JNDI is that it allows the Pega Platform to access different schemas while using only a single data source.

The following diagram illustrates the difference between a single-schema configuration and a split-schema configuration.



If you plan to change from a single-schema configuration to a split-schema configuration, do so before beginning the deployment.

Apache Ignite client-server clustering topology

You can deploy Pega Platform in client-server mode that uses Apache Ignite. Client-server mode provides greater cluster stability in large clusters and supports the ability for servers and clients to be separately scaled up. Use this mode for large production environments that consist of more than five cluster nodes or if you experience cluster instability even in clusters that contain fewer nodes. The

number of nodes in the cluster that can lead to cluster instability depends on your environment, and switching to client-server mode should be determined individually.

Client-server mode is a clustering topology that separates Pega Platform processes from cluster communication and distributed features. Clustering technology has separate resources and uses a different JVM from Pega Platform.

- Client nodes - Pega Platform nodes that perform application jobs and call the Apache Ignite client to facilitate communication between Pega Platform and the Apache Ignite servers.
- Servers - The stand-alone Apache Ignite servers that provide base clustering capabilities, including communication between the nodes and distributed features. You must have at least three Apache Ignite servers for one cluster.

For more information about enabling client-server mode, see Appendix F — Rolling restart and Apache Ignite client-server mode.

Deployment methods

You can deploy the Pega Platform either with the UI tool or from the command line. This guide includes instructions for both methods.

- Use the UI-based update deployment tool to update either the rulebase or the rulebase and the schema. You can run the update deployment tool once to update both the database schema and the rulebase, or use the command-line script to update the schema, and then run the update deployment tool to update only the rulebase.
- Command line – Run scripts to deploy the Pega Platform.

Regardless of whether you use the UI tool or the command-line scripts, you might need to edit the **setupDatabase.properties** file that controls the behavior of the following scripts:

- The **generateddl.bat** or **generateddl.sh** script generates an SQL file that your database administrator can use to apply schema changes to the database. You can run this script regardless of whether you use the update deployment tool or the command-line script.
- The **update.bat** or **update.sh** script performs the following functions:
  - Deploys the most recent version of the Pega Platform.
  - Specifies whether to generate a DDL file of changes to the database.

- Enables Kerberos authentication.

If you use the IUA to upgrade, you do not use the **update.bat** or **update.sh** script.

In-place and out-of-place updates

In an out-of-place update, an offline migration rules schema is updated, while the data schema is updated directly. The offline schema can reside either on the production database or on a second temporary database. These updates significantly minimize down time because they modify an offline schema and do not update the schemas in production. As a best practice, use out-of-place updates for split-schema configurations.

Note: Do not use the Pega Platform during an in-place update.

Update effort estimation

An update distribution image is approximately 1 GB, and up to 25% faster than running a full upgrade. The actual deployment time is influenced by four factors:

- Database structure - The update creates database indexes for an optimal runtime performance of the Pega Platform. The time taken to create these indexes is proportional to the size of the database.
- Network - The update requires a network connection to the database. Run the update on the database computer, or as close to the database on the network as possible. Do not run the update on a WAN, VPN, or any high-latency network, or network with multiple hops between the update and the database.
- Database type: Total update performance can vary as much as 25%, depending on the database type.
- Hardware performance: Choose the highest-performing computer on which to run the update. Choose enterprise-class systems with multiple CPU cores and Solid-State Drives (SSDs).

Related information

The Pega Discovery Network (PDN) at https://pdn.pega.com is Pegasystems' online documentation and information site. To access the latest documentation, use the Support menu.

- *Platform Support Guide* — The *Platform Support Guide* lists the databases, drivers and application servers supported for this release.
- Deployment guides — The PDN includes the latest installation, upgrade, and update guides.
- Release notes — The release notes include information about deploying the Pega Platform that supplement the instructions in this guide. Review the release notes before you continue. Pega Platform

  Caution: This release introduces new features and functionality that might cause compatibility issues with your existing application. You might need to take additional actions before deploying.
- Updated help files — Pegasystems provides updated help on the PDN. To obtain these updates, download the current **prhelp.war** file from the PDN.
- *Multitenancy Administration Guide* — The *Multitenancy Administration Guide* describes how to configure the Pega Platform in multitenant mode after deploying.
- *Business Intelligence Exchange User Guide* — The *Business Intelligence Exchange User Guide* describes how to install the Business Intelligence Exchange (BIX) product. BIX is included in the full distribution image, but has a separate installer.
- *System Management Application Reference Guide* — The optional System Management Application monitors and controls caches, agents, requesters, listeners, and other processing.

Differences between updates and upgrades

An update is a new distribution image that contains cumulative fixes and enhancements to the product since Pega 7.0; it is not a full product release. In contrast, upgrades are full product releases. If you need to move from any version prior to Pega 7.0, you must upgrade. To move from any Pega 7.x version to the most current release, you can either update or upgrade.

The following list identifies the major differences between updates and upgrades. For more information, see PDN > Support to determine if you can update, or if you need the full upgrade:

- Updates are smaller and faster than upgrades.
  - Because an update distribution image only contains the cumulative changes since the latest major release, the distribution image is approximately 50% smaller than a typical full upgrade distribution image. Most of the difference is because of the rules in the PRPC_Rules.jar file.

  - Both the update and upgrade consist of a series of ANT targets, executed in sequence. However, the update process omits 3 ANT targets:
    - ImportPreupgradeRuntime — This step is only required to run the newest engine, which is already present on the Pega Platform.
    - UpgradeJavaSyntax — This step upgrades snippets of Java code to be compatible with the Pega Platform.
    - RemapDatabaseTables — This step maps some Data-Admin-DB-Tables to PegaDATA, which is not necessary for the Pega Platform.
  - For smaller databases (less than 10 GB), the update process can be up to 25% faster than an upgrade, because there are fewer rules to process and fewer ANT targets to execute. For larger databases (greater than 100 GB), the update process is up to 10% faster, because the performance benefits are diminished by other long-running processes. The performance varies greatly depending on the size and structure of your database.

- Updates manage superseded hotfixes. The update process contains specialized error handling for superseded hotfixes. If your system contains a hotfix that is newer than the hotfix provided in the update, the update exits.

  For information about how to force an update, see Appendix C — Troubleshooting failed updates.

- Many updates can be reversed. You can reverse an out-of-place, split-schema update on a development system with Oracle Database, Microsoft SQL Server, or PostgreSQL. Upgrades and other updates cannot be reversed.

  For more information, see Appendix A: Reverse an update.

- Updates do not include updated help. The prhelp.war file is not included in an update. You can download the file from the update page of the PDN or use the online version.

- Updates do not include additional products. If your system includes Pega Web Mashup (formerly Internet Application Composer (IAC)) or Business Intelligence Exchange (BIX), you can update the Pega Platform, and then use the upgrade distribution image to add just the latest version of the additional products to your system.

System requirements

Ensure that your system meets the minimum requirements.

UI-based update tool (update deployment tool)

If you plan to use the UI-based update deployment tool ensure that the system on which you will update meets these minimum system requirements in addition to all other requirements:

- Windows or Linux operating system
- 1 GB minimum available memory
- 2 GB minimum disk space
- Java Platform, Standard Edition Development Kit (JDK)

Application server

The application server requires:

- RedHat JBoss systems require a Web browser — required to deploy the Pega Platform applications from the Red Hat JBoss Management Console.
- Oracle systems require an Oracle JDBC type 4 driver, such as **ojdbc7.jar**. For more information about supported drivers, see the *Platform Support Guide*.
- A supported 64-bit JDK. See the *Platform Support Guide* on the PDN for a list of supported versions.

  IBM WebSphere Network Deployment requires that the deployment manager, the node agent, and the application servers are all on the same JDK version: either JDK 1.7.0 or JDK 1.7.1.
- 1 GB minimum free disk space. You might need additional storage space for debugging and logging.

Install only the Pega Platform be installed on the application server.

Application server memory requirements

The Pega Platform runs in memory (heap) on Java Virtual Machines (JVMs). In general, all activity is distributed over multiple JVMs (nodes) on the application server.

- Standard suggested system heap size is 4 - 8 GB based on monitoring of memory usage and garbage collection frequency.

- Larger heaps are advisable if your applications allow a high number of concurrent open tasks per session or cache a large collection of transaction or reference data.
- Do not deploy the Pega Platform in an environment where the heap size exceeds the vendor-specific effectiveness limit.

Oracle and IBM JDKs use compression to minimize the cost of large heaps:

- Oracle - The compression option is labeled CompressedOOPS and is effective up to 32 GB.
- IBM - The compression option is labeled CompressedRefs and is effective up to 28 GB.

In current 64-bit JVMs, compression is enabled by default.

The host application server memory size must be at least 4 GB larger than the Pega Platform heap size to allow space for the operating system, monitoring tools, operating system network file buffering, and JVM memory size (-XMX option). The minimum host application server memory size is 8 GB:

4 GB heap + 4 GB for native memory, operating system, and buffering

If the server does not have enough memory allocated to run the Pega Platform, the system can hang without an error message. The correct memory settings depend on your server hardware, the number of other applications, and the number of users on the server, and might be larger than these recommendations.

Database server

Confirm that your database server meets the requirements in the *Pega Platform Support Guide* on the PDN.

For Oracle systems, verify that your database server includes:

- 8 GB minimum RAM
- A supported version of the JDBC4 driver for your version of the database
- 10 GB minimum initial tablespace set to auto-extend
- 50 MB logfile size — This default size is sufficient for the initial installation, but will need to be resized to run the application server workload.
- If you are using Oracle 11g, do not use the UCP (Universal Connection Pool) feature in your

database. Oracle BUG 8462305 causes a failure when an application tries to call a stored procedure. This error causes the Pega Platform to work incorrectly with a database that uses UCP. To determine if UCP is in use, check for the **ucp.jar** file in the classpath of the application server.

Database connection information

When you configure the data source resources, you need the correct database connection URL. To determine the database connection URL, obtain the following information from your database administrator:

- Connection method — Service or SID
- Host name
- Port number
- Service or SID name

When you configure the application server, enter the connection string, pega.jdbc.url. Replace items in *italics* with the values for your system:

- To connect to Oracle — Use one of the following formats:
  - jdbc:oracle:thin:@localhost:*port/service-name*
  - jdbc:oracle:thin:@localhost:*port*:*SID*
- To connect to Microsoft SQL Server —

  url="jdbc:sqlserver://*server*:*port*;DatabaseName=*database*;selectMethod=cursor; sendStringParametersAsUnicode=false"
- To connect to IBM DB2 for Linux, UNIX and Windows or IBM DB2 for z/OS — jdbc:db2://*server*:*port*/*database*
- To connect to PostgreSQL — jdbc:postgresql://*server*:*port*/*database*

For Oracle databases: Setting the JVM security parameter

If you use either UNIX or Linux, set the security to urandom to avoid a known issue with the Oracle JDBC drivers.

1. Open the configuration file or console for your application server:
   - Apache Tomcat — **setenv.bat** or **setenv.sh**
   - IBM WebSphere — Use the IBM WebSphere Administrative Console
   - IBM WebSphere Application Server Liberty Core — **jvm.options**
   - JBoss Red Hat EAP — **standalone.conf** or **standalone.conf.bat**
   - Oracle WebLogic Server — **setenv.bat** or **setenv.sh**
2. If you use either UNIX or Linux, enter the following argument to set security to urandom:- Djava.security.egd=file:///dev/urandom
3. Save the changes.

Updating from Pega 7.2.2: exporting the agent schedules for the standard Pega Platform agents

If you are updating from a version prior to Pega 7.2.2, skip this section.

If you did not use the Node Classification feature in Pega 7.2.2, skip this section.

Before you continue, export all the agent schedules that you customized for the standard Pega Platform agents. Any customizations that you made in Pega 7.2.2 are lost when you update, and you need to manually update the agent schedules after the process. You can use the exported .zip file as your reference, because it cannot be imported to the updated system.

To export the agent schedules, complete the following steps:

1. Create a product rule by clicking **Records > SysAdmin > Product > +Create**. For more information, see the product rule help.
2. On the **Contents** tab, in the **Individual instances to include** section, select the *Data-Agent-Queue* class.
3. Click **Query** to get the list of all the *Data-Agent-Queue* instances in the system.
4. Select the instances that you want to export.
5. Click **Submit**.
6. Click **Save**.
7. Click **Create product file**, and download the .zip file.

Database preparation

Before you begin preparing your database, see the *Pega Platform Support Guide* on the PDN to verify that your database is supported.

Commit hotfixes

Before you deploy, commit any uncommitted hotfixes on your system. If there are uncommitted hotfixes when you deploy, the hotfixes will be overwritten and will not be included in the updated system. For information about committing hotfixes, see the online help.

Backing up your system and database

Updating modifies both the data schema and the rules data; use a backup procedure that preserves both schemas. Before updating, back up the existing database, your applications and the system itself:

1. Verify that all rules are checked in.
2. Shut down the Pega Platform application server.
3. Use your database utilities to back up the Pega Platform database.
4. If you edited any of the following Pega Platform configuration files in the APP- INF\classes directory of an EAR deployment or the WEB-INF\classes directory of a WAR deployment, include these in the backup:
   - **prbootstrap.properties**
   - **prconfig.xml**
   - logging file: **prlogging.xml** or **prlog4j2.xml**
   - **web.xml**
   - **pegarules.keyring** or any other .keyring files

5. Back up any third-party or custom JAR files that you installed. Redeploying the Pega Platform applications might delete these from your application server.

For updates from Pega 7.3 on MSSQL: Preventing update failures by renaming the pc_work_agile.pzPvStream column

Skip this section if you are not using Microsoft® SQL Server® or if you are not updating from Pega Platform 7.3.

In Pega Platform 7.3, the capitalization for the column name pc_work_agile.pzPvStream is incorrect. The correct capitalization is pzPVStream. To prevent upgrade and update failures, run the following command from the Microsoft SQL Server Management Studio (SSMS) to rename the column:

```
EXEC sp_rename 'data-schema-name.pc_work_agile.pzPvStream',
'pzPVStream', 'COLUMN'
```

Updating multitenant systems from Pega 7.1.5 and later

If you are updating from a version prior to Pega 7.1.5, skip this section.

If you do not have a multitenant system, skip this section.

The multitenant table architecture requires an additional column, pzTenantID. Several tables are now tenant-qualified; updating the Pega Platform automatically adds the multitenant column to these tables.

SQL databases do not allow the addition of a non-null column to an existing table unless the table is empty. Therefore, if the tables contain data, updating systems on those databases displays an error "Table must be empty to add column" and the update fails. For most tables, truncating the data is acceptable; however, the pr_data_admin_product table includes important data. Pegasystems provides a script to add the pzTenantID column to the pr_data_admin_product table without losing data.

To prepare the tables, follow these steps before you update. The specific steps depend on your starting version of the Pega Platform.

1. Log in to the data schema.
2. If you are updating from Pega 7.1.7 or later, skip this step and continue at step 3; these tables already include the pzTenantID column.
   If you are updating from Pega 7.1.6 or earlier, run the following commands to truncate your tables:

   ```
   truncate table pc_schedule_task;
   truncate table pr_index_schedule_task;
   truncate table pr_data_mobile_appinfo;
   truncate table pr_data_mobile_asset;
   truncate table pr_data_mobile_build;
   truncate table pr_data_mobile_certificate;
   ```

3. Add the column to the pr_data_admin_product table without truncating the data:
   a. Navigate to the AdditionalUpgradeScripts directory:

      ```
      Pega-image/scripts/AdditionalUpgradeScripts/
      ```

   b. Run the script for your database:

      ```
      database_mt_upgrade_tables.sql
      ```

Editing the setupDatabase.properties file

Skip this section if your update meets all of the following criteria:

- You will use the update deployment tool to update.
- You will allow the update to automatically apply the schema changes and do not need to create a DDL file.
- You will not enable Kerberos authentication.

If your update does not meet all of these criteria, follow the steps in this section to edit the **setupDatabase.properties** file to configure scripts to do any or all of the following tasks:

- Update the Pega Platform. Use the **update.bat** or **update.sh** script.
- Enable Kerberos authentication. Use the **update.bat** or **update.sh** script.

- Generate a DDL file of schema changes. Use the **generateddl.bat** or **generateddl.sh** script. You can use the **generateddl.bat** or **generateddl.sh** script regardless of whether you use the update deployment tool or the command-line script to update.
- Generate user-defined functions. Use the **generateudf.bat** or **generateudf.sh** script.
- Migrate schemas. Use the **migrate.bat** or **migrate.sh** script.

To edit the **setupDatabase.properties** file:

1. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Pega-image*\scripts\setupDatabase.properties
2. Specify the properties for your system. For each property, add the appropriate value after the equal sign. See Database connection properties.
3. Save and close the file.

Database connection properties

These properties specify the settings needed to connect to the database:

- The script arguments column lists the arguments for the command-line scripts.
- The Property column lists the corresponding property in the **setupDatabase.properties** file.

Script argument	Property	Description
--driverJar	pega.jdbc.driver.jar	Path and file name of the JDBC driver.
--driverClass	pega.jdbc.driver.class	Class of the JDBC driver
--dbType	pega.database.type	Database vendor type. Enter the correct database vendor: • IBM DB2 for Linux, UNIX, and Windows: **udb** • Microsoft SQL Server: **mssql** • Oracle: **oracledate** • PostgreSQL or Enterprise DB: **postgres**
--dburl	pega.jdbc.url	The database JDBC URL. For more information, see Obtaining database connection information.
--dbuser	pega.jdbc.username	User name of the Deployment user.
--dbpassword	pega.jdbc.password	Password of the Deployment user. For encrypted passwords, leave this blank.

Script argument	Property	Description
no script argument available	pega.platform.userids.issecure	For new installations, specifies whether to deploy in secured mode. For more information, see Secured mode. Upgrades and updates automatically deploy in secured mode.
--dbSchema	rules.schema.name	In a single schema environment, sets rules schema and data schema. In a split-schema configuration, sets the rules schema only.
--dbDataSchema	data.schema.name	For split-schema configurations only, sets the data schema name.
--tempdir	user.temp.dir	Optional: The location of the temp directory. Set this location to any accessible location. For example, C:\TEMP.
--mtSystem	multitenant.system	Specifies whether this a multitenant system.

Additional update properties

The following table describes additional properties in the **setupDatabase.properties** file that you might need to edit later in the process. Do not modify these properties now.

Property	Description
bypass.pega.schema	To bypass both creating the upgrade schema and automatically generating the user-defined functions, set bypass.pega.schema to true. This implies that the upgrade DDL is already applied. To create the upgrade schema and automatically generate the UDFs, leave this property blank or set it to false.
bypass.udf.generation	If you set bypass.pega.schema to false to create the upgrade schema, but still want to bypass automatically generating the user-defined functions, set bypass.udf.generation to true.
rebuild.indexes	Rebuilds database rules indexes after the rules load to improve database access performance. If rebuild.indexes=false, you can rebuild the indexes later by running the stored procedure SPPR_REBUILD_INDEXES. The amount of time this process adds to the upgrade depends on the size of your database.
update.existing.applications	For updates from Pega 7.x, set to true to run the Update Existing Application utility to ensure that your existing applications take advantage of new functionality in the Pega Platform. Run the utility first on your development system and test the changes. Then, run the utility again on the production system. The specific actions required for your application depend on your current version. You can also run this utility later from the Designer Studio. The default setting is false.
update.applications.schema	Specifies whether to run the Update Applications Schema utility to update the cloned rule, data, work, and work history tables with the schema changes in the latest base tables as part of the update. You can also run this utility later from the **prpcUtils.bat** or **prpcUtils.sh** script, or from

Property	Description
	Designer Studio. The default setting is false.
run.ruleset.cleanup	Removes older rules. In most cases, removing older rules decreases the overall upgrade time. Running the cleanup script permanently removes rules older than the current version from which you are upgrading. For example, if you are upgrading from PRPC 6.2 SP2 (06-02-20) to 7.3.1, cleanup removes rules of version 06-01-99 and older.
reversal.schema.file.name	Schema file name to be used for reversal.
automatic.resume	If the update fails, specifies whether the system restarts the update from the step where the failure occurred. The default value is true.

Performing an out-of-place update

This section describes how to perform an out-of-place update with one database. Performing an out-of-place update is a best practice because it limits downtime and can be reversed if necessary.

Note: To convert from an existing single-schema environment to a split-schema environment, see the *Pega Platform Upgrade Guide*.

The following is a general description of the update process on an existing split-schema system with one database:

1. Disable rule creation on the rules and data schemas.
2. Unless you are using IBM DB2 for Linux, UNIX, and Windows, or IBM DB2 for z/OS, create a blank schema. On IBM DB2 for Linux, UNIX, and Windows, and IBM DB2 for z/OS, the schema is automatically created.
3. Migrate only the rules from the existing rules schema to the new blank rules schema.
4. Update the new rules schema.
5. Use the migrate script to generate the rules schema objects and establish the links between the new rules schema and data schema.
6. Shut down the existing system and then upgrade the data schema.

Disabling rule creation on the rules schema

If you are using the recommended High Availability option, you can disable rule creation on the rules schema to speed the update. If you are not using the High Availability option, skip this section.

Before you update, commit all uncommitted hotfixes. Once you begin the update, ensure that no changes to the rules, including hotfixes, are applied until after the update is complete.

1. Log in as a user with the PegaRULES:HighAvailabilityAdministrator role.
2. Navigate to **System > High Availability > HA Cluster Settings**.

3. Under **Cluster Upgrade**, select **Cluster Upgrading - Disable saving of rules**.
4. Click **Submit**.
5. Log out.

Create a new rules schema

Unless you are using IBM DB2 for Linux, UNIX, and Windows, or IBM DB2 for z/OS, create a blank schema. On IBM DB2 for Linux, UNIX, and Windows, and IBM DB2 for z/OS, the schema is automatically created when the first CREATE TABLE statement executes after the update is complete.

Migrating the rules tables with one database

Use the migrate script to migrate the rules tables and other required database objects from the existing rules schema to the new rules schema. You will also use this script later to generate and apply rules objects after the update. The Deployment user performs the migrations.

Note: Pegasystems strongly recommends that you use the migration script. The use of vendor tools to manage this step is not recommended. If you do not use the migrate script to migrate the schema, there are additional manual steps required that are not documented here.

To use the migrate script, complete the following steps:

1. Use a text editor to edit the **migrateSystem.properties** file in the scripts directory of your distribution image:

   Pega-image\scripts\migrateSystem.properties

2. Configure the source properties. See Properties file parameters for more information.

```
# Connection Information
pega.source.jdbc.driver.jar=full path/DRIVER.jar
pega.source.jdbc.driver.class=database driver class
pega.source.database.type=database vendor type
pega.source.jdbc.url=URL of database
pega.source.jdbc.username=Deployment user name
pega.source.jdbc.password=password
pega.source.rules.schema=original rules schema name
pega.source.data.schema=original data schema name
```

3. Configure the target properties. See Properties file parameters for more information. Leave the target data schema name blank:

```
pega.target.jdbc.driver.jar=full path/DRIVER.JAR
pega.target.jdbc.driver.class=database driver class
pega.target.database.type=database vendor type
pega.target.jdbc.url=database URL
pega.target.jdbc.username=Deployment user name
pega.target.jdbc.password=password
pega.target.rules.schema=temporary schema
pega.target.data.schema=
```

Note: If `pega.target.data.schema` is blank, the rules schema is used by default.

4. Configure the bulkmover directory.

```
pega.bulkmover.directory=full path to output directory
```

5. Configure a temporary directory:

```
pega.migrate.temp.directory=full path to temporary directory
```

6. Enable the operations to be performed by the utility as shown below:

```
pega.move.admin.table=true
pega.clone.generate.xml=true
pega.clone.create.ddl=true
pega.clone.apply.ddl=true
pega.bulkmover.unload.db=true
pega.bulkmover.load.db=true
```

7. Disable the operations as shown below:

```
pega.rules.generate=false
pega.rule.objects.apply=false
```

8. Save the properties file.

9. Open a command prompt, and navigate to the scripts directory.
10. Type **migrate.bat** or **./migrate.sh** to run the script.

Note: Pega Platform writes command-line output to a file in the *Pega-image*\scripts\logs directory.

Update methods

Use one of the following methods to update the database while preserving customizations:

- UI tool – The UI-based update deployment tool automatically updates the schema leaving any customizations intact. See Running the update deployment tool.

These methods require a JDBC connection to the database and can be run from any Windows, UNIX, Linux or Linux on IBM z Systems with Java 7 or later. The deployment user should perform these actions.

Note: To minimize the time required, run the update on the database server. Running the installer over a VPN, a WAN, a high-latency network, or a network with more than one hop between the installer and the database will slow the update.

Running the update deployment tool

To run the update deployment tool, complete these steps:

1. Copy the compressed distribution image to the computer that you will use to run the update.
2. Extract the contents of the compressed file into an empty directory.
3. Verify that your JAVA_HOME environment variable is the same as the PATH environment variable. If not, update the PATH environment variable with the full path to the **java.exe** executable file.
4. Open the command prompt and change the directory to the root folder where you extracted the compressed distribution image.
5. Enter the following command to launch the installer:

   ```
   # java –jar Update_Setup.jar
   ```

   The update deployment tool window opens, and the Pega Platform icon is displayed on your task bar.

6. Click **Next** to display the license agreement.
7. Review the license agreement and click **Accept**.
8. If you are resuming after a previous failed update and the **Resume Options** screen is displayed, select either **Resume** or **Start Over**.
   - If you select **Resume**, the system uses the previously entered database configuration information to resume the update from the last successful process. Skip steps 9 and 10 and continue at step 11.
   - If you select **Start Over**, continue at step 9 to reenter the configuration information.
9. Select your database type and click **Next**. The **Database Connection** screen is displayed.
10. Configure the database connection. The JDBC drivers allow the Pega Platform application to communicate with the database. Specify the new rules schema name for both the **Rules Schema Name** and **Data Schema Name** fields. For information abour valid values, see See Properties file parameters.

    Note: Some of the fields on the **Database Connection** screen are prepopulated based on the type of database you selected. If you edit these or any other fields on this screen, and then later decide to change the database type, the update deployment tool might not populate the fields correctly. If this occurs, enter the correct field values as documented below, or exit and rerun the update deployment tool to select the intended database type.

    Note: If you are resuming after a failed update, some prepopulated values might be disabled.

    - **JDBC Driver Class Name** — Verify that the prepopulated values are correct.
    - **JDBC Driver JAR Files** — Click **Select Jar** to browse to the appropriate driver files for your database type and version.
    - **Database JDBC URL** — Verify that the prepopulated value is accurate.
    - **Database Username and Password** — Enter the Deployment user name and password.
    - **Rules Schema Name** — Enter the new rules schema name.
    - **Data Schema Name** — Enter the new rules schema name.
11. Optional: Specify whether you will have your database administrator manually apply the DDL

changes to the schema. These changes include the user-defined functions (UDF) supplied by Pegasystems. By default, the update deployment tool generates and applies the schema changes to your database.

- To generate and apply the DDL outside the update deployment tool, select **Bypass Automatic DDL Application** and continue the deployment. After you complete the update, manually generate and apply the DDL and UDF. For more information, see Optional: Generating and applying DDL and Optional: Installing user-defined functions.
- To have the update deployment tool automatically apply the DDL changes and the UDF, clear **Bypass Automatic DDL Application**.

12. Click **Next**.
13. Specify whether to run the Update Applications Schema utility to update the cloned rule, data, work, and work history tables with the schema changes in the latest base tables as part of the upgrade. If you do not automatically update the applications, you can also run the update applications schema utility later from the **prpcUtils.bat** or **prpcUtils.sh** script, or from Designer Studio. For information about using the Update Applications Schema utility, see the online help.
    - Optional: For other databases, select **Update applications schema** to run the Update Applications Schema utility as part of the update.
14. Click **Next**. The **Rulebase Load** screen appears.
15. Click **Start** to load the PegaRULES database. Update logs display in the log window and are also stored in the Pega-image\scripts\logs directory.

    Note: Loading the PegaRULES database can last for several hours, depending on the network proximity to the database server. The log window might appear inactive when the update deployment tool is processing large files.
16. Click **Exit** to close the update deployment tool.

Updating from the command line

To update the rules schema from the command line, configure the **setupDatabase.properties** file and run either **update.bat** or **update.sh**. The Deployment user runs these scripts.

Note: If no additional arguments are passed to the script, the script defaults to the values of the properties set in the **setupDatabase.properties** file. See Editing the setupDatabase.properties file.

1. If you have not done so already, edit the **setupDatabase.properties** file.
   a. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Pega-image*\scripts\setupDatabase.properties
   b. Configure the connection properties. For more information about the connection properties, see Properties file parameters. Use the name of the new copy of your rules schema for both the rules.schema.name and data.schema.name properties.

```
# Connection Information
pega.jdbc.driver.jar=/path-to-the-database-JAR-file/DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment user name
pega.jdbc.password=password
rules.schema.name=new-rules-schema-name
data.schema.name=new-rules-schema-name
```

   c. Save and close the file.
2. Open a command prompt and navigate to the scripts directory.
3. Run either **update.bat** or **update.sh**.

The rulebase update can take several hours, depending on the proximity of the database to the system running the script.

Pega Platform writes command-line output to a file in the *Pega-image*\scripts\logs directory.

Generating rules schema objects

Use the migrate script to generate rules schema objects so the rules schema operates properly. Use the same **migrate.bat** or **migrate.sh** script you used to migrate the rules tables from the data schema to the new rules schema, but configure the script differently. The Deployment user performs the migrations.

Note: Pegasystems strongly recommends that you use the migration script. The use of vendor tools to manage this step is not recommended. If you do not use the migrate script to migrate the schema, there are additional manual steps required that are not documented here.

Follow these steps to generate rules schema objects:

1. Use a text editor to edit the **migrateSystem.properties** file in the scripts directory of your distribution image:

   Pega-image\scripts\migrateSystem.properties

2. Keep most of the values you used when you migrated the rules tables. Verify these values and change them if necessary:

   ```
   pega.target.data.schema=original data schema
   pega.target.rules.schema=new rules schema
   pega.source.rules.schema=new rules schema
   ```

3. Disable the move.admin.table property:

   ```
   pega.move.admin.table=false
   ```

4. Set the following operational properties to false because you are not migrating the rules schema again:

   ```
   pega.clone.generate.xml=false
   pega.clone.create.ddl=false
   pega.clone.apply.ddl=false
   pega.bulkmover.unload.db=false
   pega.bulkmover.load.db=false
   ```

5. Set the following properties to true to generate and apply rules objects:

   ```
   pega.rules.objects.generate=true
   pega.rules.objects.apply=true
   ```

6. Save the properties file.

7. Open a command prompt, and navigate to the scripts directory.

8. Type **migrate.bat** or **./migrate.sh** to run the script.

Note: Pega Platform writes command-line output to a file in the *Pega-image*\scripts\logs directory.

Next step

Continue at Upgrading the data schema from the command line.

Updating the data schema

The Deployment user runs the update script to update the data schema:

1. Shut down your system.
2. Edit the **setupDatabase.properties** file.
   a. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Pega-image*\scripts\setupDatabase.properties
   b. Configure the connection properties. See Properties file parameters for more information:

```
# Connection Information
pega.jdbc.driver.jar=/path-to-the-database-JAR-file/DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment user name
pega.jdbc.password=password
rules.schema.name=new rules schema
data.schema.name=existing data schema
```

   c. Optional: If you are repeating a failed update, configure the resume property:
      - To resume the update from the last successful step, set `automatic.resume=`**true**.
      - To restart the update from the beginning, set `automatic.resume=`**false**.
   d. Save and close the file.
3. Disable the Pega Platform access to the data schema.
4. Open a command prompt, and navigate to the scripts directory.
5. Type **update.bat** or **./update.sh** for Linux, passing in the --dataOnly argument and true parameter, for example:

```
update.bat --dataOnly true
```

Note: The Pega Platform writes command-line output to a file in the *Pega-image*\scripts\logs directory. This process also generates a **Reverse_*timestamp*.xml** file in the scripts\logs directory. A new file is generated each time the data schema is updated.

Post-update configuration

The specific tasks you must perform depends on the version from which you are updating.

For Docker, multiple VMs, or multiple NICs: Setting the public address

If the cluster is set up in Docker, uses separate virtual machines (VMs), or multiple network interfaces (NICs), set the public address in the **prconfig.xml** file for each Pega Platform node.

1. Open the **prconfig.xml** configuration file that is located in the prweb/WEB-INF/classes subdirectory of the application server directory. For more information, see the PDN article *How to change prconfig.xml file settings*.
2. Modify the **prconfig.xml** file. Add the following setting to set the public address:

   ```
   <env name=" identification/cluster/public/address" value="IP address" />
   ```

   For example, if the IP address of the computer on which you run the Pega Platform node is 10.254.34.210, add the following setting:

   ```
   <env name=" identification/cluster/public/address"
   value="10.254.34.210" />
   ```

   The new setting controls the address that is used by the Pega Platform node.
3. Save and close the **prconfig.xml** file.
4. Repeat steps 1 to 3 for the remaining nodes.

Reconfiguring the application server

To use the updated rules schema, you must reconfigure the application server. The process is different for each application server:

1. Replace all references to the old rules schema with the new rules schema name (for example, OldRULES -> NewRULES)

2. Remove the old Pega Platform application and WAR or EAR file, and replace it with the new WAR or EAR file in the \archives\ directory.

After you reconfigure the application server, restart the Pega Platform using the new system.

For updates from Pega 7.1.6 and earlier — Redeploy applications

Skip this section if you are updating from Pega 7.1.7 or later. Follow the instructions in this section to redeploy the applications only if you are updating from Pega 7.1.6 or earlier.

Remove the existing Pega Platform applications from your application server and deploy the new core applications in the *Pega-image*\ directory. The core applications include:

- **prweb.war** (or **prpc_j2ee14_ws.ear**) — Designer Studio
- **prsysmgmt.war** — System Management Application (SMA)
- **prhelp.war** — on-line help

Note: The **prhelp.war file** is not included in the update distribution image. Download the file from the PDN at Support > Downloads > Pega Platform Update Software.

Redeploy all Pega Platform applications:

- prweb — DesignerStudio
- prhelp — help
- prsysmgmt — System Management Application
- Industry applications
- Custom applications

Note: Do not start the redeployed applications while the rulebase update process is running in the update deployment tool. By default, your application server might start the applications automatically when they are deployed. If you deploy and start the applications before creating the database, the applications generate an error and fail to start. This error is not harmful, and you can restart the application successfully when the database is available.

Start the Pega Platform applications

Ensure that the application server is running and start the prsysmgmt, prweb, and prhelp applications.

WebSphere

To start the applications:

1. In the IBM WebSphere administrative console, select **Applications > Application Types > WebSphere enterprise applications**.
2. Select the three applications and click **Start**.

   The application status turns green and the system displays a success message.

Logging in to Pega Platform and changing the administrator password

To test the deployment and index the rules, log in to Pega Platform web application.

1. Navigate to the PRServlet URL, replacing the *server* and *port* values with your specific values.

   ```
   http://server:port/prweb
   ```

2. Use the following credentials to log in:

   - User ID — **administrator@pega.com**
   - Password — **install**

   After logging in, Pega Platform indexes the rules in the system to support full-text search. During the index process, there might be a delay in the responsiveness of Pega Platform user interface. The process usually takes from 10 to 15 minutes to complete depending on your system configuration.

   If the index process ends without generating an error message, the deployment is successful.

3. Immediately after the index process completes, change the administrator password. Because the

default password is widely published, it is important to protect your system after an installation by changing the password. The new password must be at least 10 characters long.

If the system does not prompt you to change your password, follow these steps:

a. From the **Operator Menu** located to the right of the Designer Studio header, select the **Profile**.

b. Click **Change Password**.

c. Verify the **Current Password**, and then enter and confirm the **New Password**.

d. Click **Save**.

Manually building Elasticsearch indexes

For updates from Pega 7.1.6 and earlier, manually build the Elasticsearch indexes and configure the search index host node settings as soon as possible after updating the Pega Platform.

The Pega Platform supports full-text search for rules, data instances, and work objects. By default, search indexing is enabled and indexing starts when you start the application server after updating the Pega Platform. The first node that starts after update becomes the default initial search node. The default index directory is PegaSearchIndex in your temporary directory.

After the search indexes are completely built, you can change the default settings. Do not stop or bring down the default node until the search indexes build completely. The Search Landing Page displays the status.

Note: Starting in Pega 7.1.7, the underlying platform for full-text search transitioned from Lucene to Elasticsearch. Elasticsearch provides a more robust, fault-tolerant search capability and does not require manual configuration of switchover activities. Existing search customizations through Pega Platform APIs are intact and used exactly the same way with Elasticsearch; only the search query generation changes from Lucene to Elasticsearch.

Follow these steps to build the indexes:

1. Check your directory sizes. Ensure that the directories for all Elasticsearch host nodes have sufficient free space to hold the Elasticsearch indexes.
   - For updates from Pega 7.1.6 and earlier, ensure that the host node directories have sufficient free space to hold the Elasticsearch indexes. Elasticsearch indexes are approximately three times the size of the Lucene indexes.

- For updates from Pega 7.1.6 and earlier, ensure that the directory for the initial host node has sufficient space to initially hold both the Lucene index and the Elasticsearch index.

2. Use the prpcUtils tool to reindex the rules:
   a. On the node that you want to index, open the **prpcUtils.properties** file in the Pega_HOME\scripts\utils directory.
   b. Configure the connection properties. For more information about parameter values, see Properties file parameters.

```
# Connection Information
pega.jdbc.driver.jar=/path-to-the-database-JAR-file/DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment username
pega.jdbc.password=password
```

   c. Optional. Configure a directory to store the new indexes; by default, indexes are created in the directory specified in the `user.temp.dir` property.

```
indexing.indexdirectory=full-path/index/
```

   d. Configure the indexing type parameter in the **SETTINGS FOR FULL TEXT INDEXER TOOL** section; leave all other indexing parameters commented out:

```
indexing.indextype=Rule
```

   e. Save and close the **prpcUtils.properties** file.
   f. Run the **prpcUtils.bat** or **prpcUtils.sh** script to reindex the rules. For example:

```
prpcUtils.bat indexing
```

3. Repeat step 2 to reindex the data files. Set the index type to Data:

```
indexing.indextype=Data
```

4. Repeat step 2 to reindex the work files. Set the index type to Work:

```
indexing.indextype=Work
```

5. Use Designer Studio to delete the existing index nodes:
   a. Open the **Designer Studio > System > Settings> Search** landing page and expand **Search Index Host Node Setting**.
   b. Click the **X** to the right of each node to delete all existing nodes.
   c. Click **Submit**.
6. Use Designer Studio to add the host nodes. The system replicates the indexes on the new nodes.

   Note:
   - Configure a minimum of two Elasticsearch host nodes. Pegasystems recommends that you configure a minimum of three nodes for maximum fault tolerance. You might need more than three nodes depending on the size of your cluster.
   - You can specify that a node is either always an index host node or that it never becomes an index host node even if it is the first node that is started after installation using the `-Dindex.directory` JVM setting. To specify that a node is always an index host node specify the directory name. To specify that a node is never an index host node, leave this setting blank. For more information about configuring index host nodes, see *Managing Elasticsearch index host nodes outside of the Search* landing page on the PDN.

   a. Open the **Designer Studio > System > Settings> Search** landing page and expand **Search Index Host Node Setting**.
   b. Enter the information for the primary host node. The first node you enter is the node on which Elasticsearch indexes will be built.
      i. Enter the Search index host node ID on which you built the indexes.

         For example:

         ```
         /dsk01/tomcat7/system7/work/Catalina/localhost/prweb/PegaSearchIndex.
         ```

      ii. In the Search index file directory, enter the directory in which prpcUtils saved the indexes.
   c. Repeat step 5.b to add any needed additional host nodes.
   d. Verify the **Search Index Host Node ID** and the **Search Index File Directory**.

e. Expand **Automated Search Alerts**, and enable **Automatically Monitor Files**.

f. Click **Submit** to save the settings.

7. To enable communication between Elasticsearch host nodes in the cluster, open a TCP port in the range 9300-9399 on each node. (By default, Elasticsearch uses port 9300.) These ports are used for internal node-to-node communication only, and should not be externally accessible.

Cleaning up unused tables

Pegasystems recommends that you drop unused rules tables in the data schema after deploying a split-schema. If you have only one database, also drop unused data tables in the rules schema.

1. Verify that you have the correct privileges to view and edit the **Optimize Schema** landing page. Set these to `true`:

   - ViewAndOptimizeSchema
   - Dynamic System Setting (DSS) databases/AutoDBSchemaChanges
   - ViewSchemaOnImport
   - SchemaImport

   See the PDN article *How to set privileges to restrict database schema changes generated from Process Commander* (https://pdn.pega.com/how-set-privileges-restrict-database-schema-changes-generated-process-commander) for more information.

2. To open the optimize schema wizard, click **Designer Studio > System > Database > Optimize Schema.**

3. Select the PegaDATA database.

4. Click **view the unused tables** to display a list of Pega Platform tables without class mappings. Either select the ones you want to delete and click **Proceed with Changes** to have Pega Platform drop the tables, or drop them manually in your database.

5. Repeat steps 3 and 4 for the PegaRULES database.

Upgrading from Pega 7.2.2 or earlier: Updating access role names to enable notifications

When upgrading from any release prior to Pega 7.2.2, you must update all the user access role names of an application with the following classes so that users can receive notifications.

- *Data-Notification-Parameters*
- *Pega-Notification*
- *Data-Notification-Recipient*
- *Data-Preference-Operator*

Tip: To save time, you can clone any access role name that contains the preceding classes and assign it to application users instead of updating the access role names manually. For more information on how to clone a access role name, see the *Access Role form - Completing the Role* tab help topic.

To update access role names:

1. In the Records Explorer, click **Security > Access Role Name**.
2. Open the access role name that needs to be updated.
3. Click the **Plus** icon to open the **Add Access Role Object** dialog box.
4. In the **Class** field, enter the class name that you want to add to the access role name.
5. Under Access Control, enter **5** in all the fields to provide access to this access role name.
6. Click **Save**.
7. Perform steps 3 through 6 for each of the remaining classes.

Enabling access to environmental information

Prior to Pega 7.3, all roles included access to environmental information for the current node. This information can include version numbers of third-party platforms and JVM information. This access appears as a flaw in some security audits. With Pega 7.3, the new *@baseclass.pxViewSystemInfo* privilege controls access to environmental information. Only the PegaRULES:SysAdm4 role has this privilege by default.

After updating from any version prior to Pega 7.3, add the *@baseclass.pxViewSystemInfo* privilege to all system administrator roles that need access to environmental information.

1. Click **Designer Studio > Org & Security > Tools > Security > Role Names**.
2. In the pop-up window that displays roles, click the role that you want to update.
3. In the Designer Studio click the *@baseclass* class in the **Access Class** column.
4. In the **Privileges** section, click the **Plus** icon and select the *pxViewSystemInfo* privilege in the **Name** column.
5. Enter **5** for the production level in the **Level** column. Production level 5 provides the highest security.
6. Click **Submit**.
7. Repeat steps 1 - 6 for each role that requires modification.

Enabling operators

After deploying the Pega Platform in secured mode, enable new out-of-the-box operator IDs. For more information about secured mode, see Appendix E — Secured mode.

Note: When you deploy the Pega Platform in secure mode there are no changes to any existing administrator@pega.com operator in the target database. However, if there is no administrator@pega.com operator record, the deployment creates the operator.

To enable operators, follow these steps:

1. Select **Designer Studio > Org & Security > Authentication > Operator Access**.
2. In the **Disabled operators** list, select the operators to enable. The following standard operators are installed but disabled by default. Enable only those operators you plan to use:
   - Batch@pega.com
   - DatabaseAdmin@pega.com
   - ExternalInviteUser
   - IntSampleUser
   - PRPC_SOAPOper
   - PortalUser@pega.com
   - UVUser@pega.com
   - External
3. Click **Enable selected**.
4. Click **Confirm**.

When these standard operators first log on, they are required to change their passwords.

Enabling server-side screen captures for application documents

Regardless of which application server platform you use, you must set up a Tomcat server to support taking and storing screen captures on a server rather than on a client. By taking and storing screen captures on a server, you avoid client-side limitations, such as browser incompatibilities or client software requirements.

Tip: As a best practice, virtually install Tomcat and deploy the **prScreenShot.war** file on the same server that is running the Pega Platform. Otherwise, use a standalone Linux or Windows server. If you use a Linux server, you must include the following components:

- fontconfig
- freetype

- libfreetype.so.6
- libfontconfig.so.1
- libstdc++.so.6

You can include screen captures in an application document that is generated by the Document Application tool. Screen captures provide stakeholders with a realistic picture of an application's user interface. Install a PhantomJS REST server to include screen captures in an application document.

1. Download the following WAR file: **Pega_DistributionImage\Additional_Products\PhantomJS\prScreenShot.war**
2. Deploy the WAR file on a Tomcat server.
3. Update the **tomcat-users.xml** file to add the following role and user. This file is located at \apache-tomcat-XX\conf\ tomcat-users.xml.
   - Set the role:

     ```
     <role rolename="pegascreencapture" />
     ```

   - Set the user:

     ```
     <user username="restUser" password="rules"
     roles="pegascreencapture" />
     ```

4. Start the Tomcat server. The service is hosted at `http://IPaddress:port/prScreenShot/rest/capture`, where *IPaddress* is the address of the system where Tomcat is hosted, and *port* is the port on which the service is deployed.
5. Log in to your Pega Platform application and make the following updates:
   a. Update the Data-Admin-System-Setting instance *Pega-AppDefinition - CaptureScreenshotsResourcePath* with the URL of the service, for example, `http://10.224.232.91:8080/prScreenShot/rest/capture`.
   b. Update the Data-Admin-Security-Authentication profile instance CaptureScreenshotsAuthProfile with the user that you created in step 3.

Configuring PhantomJS REST server security for including screen captures in an application document

To ensure a secure installation of the Pega Platform, you must enable the PhantomJS REST server to take and store server-side screen captures. In application documents generated by the Document Application tool, screen captures provide stakeholders with a realistic picture of the application's user interface.

1. Obtain the SSL certificate from the Pega Platform administrator.
2. Add the SSL certificate to the list of trusted certificates:
   a. Double-click the certificate.
   b. Click **Install certificate** to start the **Certificate Import** wizard.
   c. Click **Next**, and select **Place all certificates in the following store**.
   d. Click **Browse**, select **Trusted Root certificate**, and click **OK**.
   e. Click **Next**, and then click **Finish** to complete the wizard.
3. Add the certificate to the truststore of the JVM on which the REST server is installed:
   a. Open a command prompt.
   b. Change the root directory to the security folder in the Java installation folder.

      For example, `C:\Program Files (x86)\Java\jre7\lib\security`.
   c. Run the following command:
      ```
      keytool -keystore cacerts -importcert -alias certificate alias -
      file certificate name
      ```
   d. When prompted, enter the password for the cacerts keystore. The default password is `changeit`.

Updating from Pega 7.2.2: customizing the agent schedules for the standard Pega Platform agents

If you are deploying from a version prior to Pega 7.2.2, skip this section.

If you did not use the Node Classification feature in Pega 7.2.2, skip this section.

You need to manually update all the agent schedules that you customized in Pega 7.2.2 for standard Pega Platform agents. You can update the agent schedules after starting a node with a node type, when the agent schedule is re-created.

1. Click **Designer Studio > System > Operations > Node Classification**.
2. On the **Agents** tab, in the **Associated with Node type** column, click the name of the node type that is associated with the agent for which you want to update an agent schedule.

   Note: If the agent schedule has not been generated, you can create it by clicking **+Create** in the **Agent schedule** column.
3. In the agent schedule form, modify any settings that need to be updated. For more information, see the help for the agent schedule data instances.
4. Click **Save**.

Appendix A — Properties files

The Pega Platform properties files include several database-specific properties.

This list of supported values is organized by database:

IBM DB2 for Linux, UNIX and Windows

- JDBC driver JAR file — **db2jcc4.jar**
- Database driver class — com.ibm.db2.jcc.DB2Driver
- Database vendor type — udb
- JDBC URL — url="jdbc:db2://*host:port/dbname*

Microsoft SQL Server

- JDBC driver JAR file — **sqldbc.jar**
- Database driver class — com.microsoft.sqlserver.jdbc.SQLServerDriver
- Database vendor type — mssql
- JDBC URL — url="jdbc:sqlserver://*host:port*;databaseName=*dbName*;SelectMethod=cursor;SendStringParametersAsUnicode=false"

Oracle

- JDBC driver JAR file — **ojdbc7.jar**
- Database driver class — oracle.jdbc.OracleDriver
- Database vendor type — oracledate
- JDBC URL — Use either the service name or the SID:
  - url="jdbc:oracle:thin:@*host:port/service-name*"
  - url="jdbc:oracle:thin:@*host:port:SID*"

PostgreSQL

- JDBC driver JAR file
  - PostgreSQL 9.3 — **postgresql-9.3.1103.jdbc3.jar**
  - PostgreSQL 9.4 — **postgresql-9.4.1207.jre6.jar**, **postgresql-9.4.1207.jre7.jar**, or **postgresql-9.4.1207.jar**
  - PostgreSQL 9.5 — **postgresql-9.4.1211.jre7.jar** or **postgresql-9.4.1211.jar**
  - PostgreSQL 9.6 — **postgresql-9.4.1212.jre7.jar** or **postgresql-9.4.1212.jar**
- Database driver class — org.postgresql.Driver
- Database vendor type — postgres
- JDBC URL — url="jdbc:postgresql://*host:port/dbname*"

Appendix B — Reverse an update

You can revert to a previous release after performing an out-of-place, split-schema update on a Pega Platform development system with Oracle, Microsoft SQL Server, or PostgreSQL.

Reversing an update requires the original (pre-update) rules schema, which is available only if you perform an out-of-place update.

Limitations

Reversing an update is not supported for:

- In-place updates
- Single-schema systems
- Multitenancy systems
- Production environments
- IBM DB2 for Linux, UNIX, and Windows or IBM DB2 for z/OS
- Multi-hop updates: For example,
  - You can:
    - Update from Pega 7.1.5 to Pega 7.1.7.
    - Reverse the update to return Pega 7.1.5.
  - You cannot:
    - Update from Pega 7.1.5 to Pega 7.1.6 (first hop).
    - Update again from Pega 7.1.6 to Pega 7.1.7 (second hop).
    - Reverse the update back to Pega 7.1.5.

Update and reversal files

To reverse an update, run the **reverse.bat** or **reverse.sh** script on the pre-updated rules schema to revert the changes to the data schema.

Updating the data schema creates a **Reverse_*timestamp*.xml** file. The **reverse.bat** or **reverse.sh** script uses the **Reverse_*timestamp*.xml** file to re-create the data schema. If you update your system more than once, there might be multiple versions of the **Reverse_*timestamp*.xml** file. The **setupDatabase.properties** file specifies the correct **Reverse_*timestamp*.xml** file.

The reverse process restores database functions, procedures, triggers, and views in the data schema. New or altered columns, constraints, indexes, and tables are left intact. The process creates the **CLI-Reverse-Log-*timestamp*.log** file in the scripts/logs directory.

If new *Data-* instances (including *Work-* instances) are created by using data objects modified by the update process, the new instances might not function properly after reversal.

The table below describes the reverse behavior for some database objects that are either added or modified (before or after an update), or added or modified by the product during the update process.

Object Type	How created	Added or Modified	Status after Reversal
Tables, Columns, Indexes	--	--	Ignored
Procedures, Triggers, Views	--	Added or Modified	Restored
	--	Added	Ignored
	--	Modified	Restored
	--	Added	Dropped
	--	Modified	Restored
Functions	Manually (created by you)	Added or Modified	Restored
	Pega Platform shipped	Added or Modified	Ignored
	Manually	Added	Ignored
	Manually	Modified	Restored
	Pega Platform shipped	Added or Modified	Ignored
	Pega Platform shipped	Added or Modified	Ignored
Data instances	Pega Platform shipped	Added or Modified	Restored
	Pega Platform shipped	Added	Ignored
	Pega Platform	Modified	Ignored. If the same instance is also modified during the

Object Type	How created	Added or Modified	Status after Reversal
	shipped		update, it will be restored.
	Pega Platform shipped	Added	Deleted
	Pega Platform shipped	Modified	Restored
	Manually	Added or Modified	Ignored

Reversing an update

To reverse the update:

1. Check the scripts/log directory for the presence of multiple **Reverse_*timestamp*.xml** files. Note the file name with the latest time stamp.
2. If you have not done so already, edit the **setupDatabase.properties** file to configure the reversal.
   a. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Pega-image*\scripts\setupDatabase.properties
   b. Configure the connection properties. For more information about the connection properties, see Properties file parameters.

```
# Connection Information
pega.jdbc.driver.jar=/path-to-the-database-JAR-file/DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment user name
pega.jdbc.password=password
rules.schema.name=new-rules-schema-name
data.schema.name=data-schema-name
```

   c. If there are multiple versions of the **Reverse_*timestamp*.xml** file, specify the file name from step 1 in the reversal.schema.file.name property, for example:

```
reversal.schema.file.name=Reverse_2016-06-18-23:59:59.xml
```

   d. Save and close the file.

3. In the scripts directory of the original rules schema, run the **generateDDL.bat** or **generateDDL.sh** script to generate the reverse create and drop statements.
4. In the scripts directory of the original rules schema, run the **reverse.bat** or **reverse.sh** script. If there are any errors, review the information in the scripts\logs\CLI-Reverse-Log-*timestamp*.log file.
5. Reconfigure the application server:
    a. Restart the application server.
    b. Verify that the **prweb** application is configured to use the correct rules and data schema names.
    c. Deploy the **prweb** application.
6. Check the version of the Pega Platform on the Designer Studio log-in screen.

Appendix C — Optional: Generating and applying DDL

Skip this section if you will have the update deployment tool automatically apply the DDL or if you plan to use the JCL to generate a DDL file on an IBM DB2 for z/OS system.

Manually generating and applying DDL changes must be done in each step of the deployment. Some steps use the generateddl script. Other steps use the migrate script. These scripts write platform-specific DDL to a file. You can then view and edit the file or directly apply it by using database management tools. Both scripts work identically and accept the arguments noted in Editing the setupDatabase.properties file.

Manually generating and applying the DDL requires you to use the generateDDL script several times. For example, this is the generic process for updating a split-schema system with one or two databases:

1. Optional: If you are updating out-of-place, migrate the rules:
   a. Clone the DDL.
      i. Edit the **migrateSystem.properties** file to set the source schema names:

```
pega.source.rules.schema=original rules schema name
pega.source.data.schema=original data schema name
```

      ii. Edit the **migrateSystem.properties** file to set the target schema names. The settings depend on whether you have one database or two databases:

- One database:

```
pega.target.rules.schema=new rules schema name
pega.target.data.schema=new rules schema name
```

- Two databases:

```
pega.target.rules.schema=temporary schema name
pega.target.data.schema=temporary schema name
```

      iii. Edit the **migrateSystem.properties** file to create the DDL:

```
pega.move.admin.table=true
pega.clone.create.ddl=true

pega.clone.apply.ddl=false
pega.bulkmover.unload.db=false
pega.bulkmover.load.db=false
pega.rules.objects.generate=false
pega.rules.objects.apply=false
```

iv. Run the **migrate.sh** or **migrate.bat** script to create the DDL.

b. Have the database administrator apply the DDL.

c. Populate the tables.

i. Leave the source and target schema properties as in step 1a.

ii. Edit the **migrateSystem.properties** file to populate the table:

```
pega.move.admin.table=true
pega.bulkmover.unload.db=true
pega.bulkmover.load.db=true

pega.clone.create.ddl=false
pega.clone.apply.ddl=false
pega.rules.objects.generate=false
pega.rules.objects.apply=false
```

iii. Run the **migrate.sh** or **migrate.bat** script to populate the table.

2. Update the rules schema and apply the DDL for the rule schema changes:

a. Create the DDL of changes to the rules schema. For more information about the scripts, see Generating the DDL file.

i. Edit the **setupDatabase.properties** file to set the rules and data schema names:

- One database:

```
rules.schema.name=new rules schema name
data.schema.name=new rules schema name
```

- Two databases:

```
pega.target.rules.schema=temporary schema name
pega.target.data.schema=temporary schema name
```

ii. Run the **generateddl.bat** or **generateddl.sh** script.

b. Have the database administrator apply the DDL.

c. Use the command line to update the rules schema. For more information, see Updating from the command line.

i. Edit the **setupDatabase.properties** file to bypass the schema upgrade update because the DDL is already applied:

```
bypass.pega.schema=true
```

ii. Leave the rules and data schema names as in step 2a.

iii. Run the **update.bat** or **update.sh** script.

3. Migrate the changes to the new rules schema; create rules schema objects, and create links between the new rules schema and the data schema. The steps differ depending on whether you have one or two databases.

- One database

a. Generate rules objects from the rules schema and create links between the new rules schema and the data schema:

i. Edit the **migrateSystem.properties** file to set the source and target schema properties:

```
pega.source.rules.schema=new rules schema
pega.source.data.schema=new rules schema
pega.target.rules.schema=new rules schema
pega.target.data.schema=original data schema
```

ii. Edit the **migrateSystem.properties** file to generate the DDL that is needed to create the rules objects:

```
pega.rules.objects.generate=true

pega.move.admin.table=false
pega.clone.create.ddl=false
pega.clone.apply.ddl=false
pega.bulkmover.unload.db=false
pega.bulkmover.load.db=false
pega.rules.objects.apply=false
```

iii. Run the **migrate.sh** or **migrate.bat** script to create the DDL.

b. Give the DDL to the database administrator to apply the rules objects.

- Two databases

a. Clone the DDL.

i. Edit the **migrateSystem.properties** file to set the source and target schema properties:

```
pega.source.rules.schema=temporary schema name
pega.source.data.schema=temporary schema name
pega.target.rules.schema=new rules schema
pega.target.data.schema=original data schema
```

ii. Edit the **migrateSystem.properties** file to create the DDL:

```
pega.clone.create.ddl=true

pega.move.admin.table=false
pega.clone.apply.ddl=false
pega.bulkmover.unload.db=false
pega.bulkmover.load.db=false
pega.rules.objects.generate=false
pega.rules.objects.apply=false
```

iii. Run the **migrate.sh** or **migrate.bat** script to create the DDL.

b. Give the DDL to the database administrator to apply.

c. Populate the tables.

i. Leave the source and target schema properties as in step 3a.

ii. Edit the **migrateSystem.properties** file to populate the table:

```
pega.bulkmover.load.db=true
pega.rules.objects.generate=true
pega.bulkmover.unload.db=true

pega.move.admin.table=false
pega.clone.create.ddl=false
pega.clone.apply.ddl=false
pega.rules.objects.apply=false
```

iii. Run the **migrate.sh** or **migrate.bat** script to populate the table.

d. Give the DDL to the database administrator to apply the rules objects.

4. Update the data schema and apply the DDL for the data schema changes:

a. Create the DDL of changes to the rules schema. For more information about the scripts, see Generating the DDL file.

i. Edit the **setupDatabase.properties** to set the rules and data schema names:

```
rules.schema.name=new rules schema
data.schema.name=original data schema
```

ii. Run the **generateddl.bat** or **generateddl.sh** script.

b. Have the database administrator apply the DDL to the data schema.

c. Use the command line to update the data schema. Follow the instructions in Updating the data schema.

i. Edit the **setupDatabase.properties** file to bypass the schema update because the DDL is already applied:

```
bypass.pega.schema=true
```

ii. Run the **update.bat** or **update.sh** script with the --dataOnly argument and true parameter, for example:

```
update.bat --dataOnly true
```

Generating the DDL file

Follow these steps to generate a DDL file:

1. Edit the **setupDatabase.properties** file.
   a. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Pega-image*\scripts\setupDatabase.properties
   b. Configure the connection properties. For more information about parameter values, see Properties file parameters.

```
# Connection Information
pega.jdbc.driver.jar=\path-to-the-database-JAR-file\DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment username
pega.jdbc.password=password
rules.schema.name=rules-schema-name
data.schema.name=data-schema-name
```

   c. Save and close the file.
2. At a command prompt, navigate to the *Pega-image*\scripts directory.
3. Type **generateddl.bat** or **generateddl.sh** and pass in the required --action argument:

```
#generateddl.bat --action upgrade
```

If you do not specify an output directory, the script writes the output to the default directory:

```
Pega-image\schema\generated\
```

Note: The output directory is deleted and re-created each time the generateddl script runs. To save a copy of the DDL, rename the directory before you run the script.

Applying the DDL file

Before you continue, have your database administrator follow these general steps to apply the schema changes; these schema changes can include changes to user-defined functions:

1. Review the DDL file in the output directory and make any necessary changes. The default directory is:

   Pega-image\schema\generated\database*database_type*

   where *database_type* is udb, mssql, oracledate, postgres, or db2zos

2. Apply the DDL file.
   a. Register the DDL file with the database:
      - For Oracle, PostgreSQL, and IBM DB2 for Linux, UNIX, and Windows, register the .jar file with the database.
      - For Microsoft SQL Server, enable the user-defined functions and register the C# .cs code files with the database.
   b. Apply the **CREATE FUNCTION** DDL.

Note: The output directory is deleted and re-created each time the generateddl script runs. To save a copy of the DDL, rename the directory before you rerun the script.

Appendix D — Installing user-defined functions

The user-defined functions (UDFs) enable the Pega Platform to read data directly from the BLOB without creating and exposing columns. Skip this section if you installed the UDFs as part of the update.

There are several ways you might have bypassed generating and installing the UDFs during the update:

- Setting either `bypass.pega.schema=true` or `bypass.udf.generation=true` in the **setupDatabase.properties** file
- Setting `pega.target.bypass.udf=true` in the **migrateSystem.properties** file
- Selecting **Bypass Automatic DDL Application** from the update deployment tool during the update

Before you install the UDFs, verify that you have the appropriate user permissions.

For more information about user permissions, see your Pega Platform installation guide.

1. Edit the **setupDatabase.properties** file.
   a. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Pega-image*\scripts\setupDatabase.properties
   b. Configure the connection properties. For more information about parameter values, see Properties file parameters.

```
# Connection Information
pega.jdbc.driver.jar=\path-to-the-database-JAR-file\DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment user name
pega.jdbc.password=password
rules.schema.name=rules-schema-name
data.schema.name=data-schema-name
```

   c. Save and close the file.

2. On the rules schema, navigate to the *Pega-image*\scripts directory and run the following commands to remove any partially installed UDFs:

```
DROP FUNCTION rules-schema-name.pr_read_from_stream;
DROP FUNCTION rules-schema-name.pr_read_decimal_from_stream;
DROP FUNCTION rules-schema-name.pr_read_int_from_stream;
```

3. Optional: If you have a split-schema, on the data schema, navigate to the *Pega-image*\scripts directory and run the following commands:

```
DROP FUNCTION data-schema-name.pr_read_from_stream;
DROP FUNCTION data-schema-name.pr_read_decimal_from_stream;
DROP FUNCTION data-schema-name.pr_read_int_from_stream;
```

4. From the *Pega-image*\scripts directory, run the **generateudf.bat** or **generateudf.sh** script with the `--action install` argument, for example:

```
generateudf.bat --action install --dbType database_type
```

   Where the database type is mssql, oracledate, udb, db2zos, or postgres.

Editing the setupDatabase.properties file to bypass DDL generation

After your database administrator applies the changes to your database, configure the **setupDatabase.properties** file to bypass applying a schema that already exists. Reapplying an existing schema would cause the update to fail.

To edit the **setupDatabase.properties** file:

1. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Pega-image*\scripts\setupDatabase.properties
2. Set the property `bypass.pega.schema=true`.
3. Save and close the file.

Appendix D — Troubleshooting failed updates

Use the information in this section to troubleshoot update errors. Update error logs are displayed in the update deployment tool window and are also stored in the Pega-image\scripts\logs directory.

Resuming or restarting after a failed update

If the update fails, you can opt to either resume or start over:

- **Resume** — The system uses the previously-entered configuration information to resume a failed update from the last successful step. This is the default behavior.
- **Start Over** — The system discards all previously-entered configuration information, drops the database schema, and starts the update from the beginning.

To resume or restart the update:

1. Review the update failure message for information about the source of the error. Use the information in the error message to correct the error before you continue.
2. Depending on the update method that you used, do one of the following actions:
   - If you used the update deployment tool, the select either **Resume** or **Start Over** when the system displays the **Resume Options** screen.
   - If you used the command-line script, set the automatic.resume property in the **setupDatabase.properties** file:
     - Set `automatic.resume=true` to resume the update from the last successful step.
     - Set `automatic.resume=false` to start over.
3. Repeat the update. Use the same procedure that you used for the initial update.

Error: System contains hotfixes generated after this release was built — forcing an update

If your system contains hotfixes that are newer than the update you are attempting to apply, the update fails with the error “This update cannot be applied, the system contains hotfixes generated after this release was built.” The hotfixes are listed in the log file. Use the force update option to bypass this error.

Forcing an update by using arguments

To force an update by using arguments:

1. Verify the settings in the **setupDatabase.properties** file. For information about the properties, see Editing the setupDatabase.properties file.
2. At a command prompt, navigate to the *Pega-image*\scripts directory.
3. Run one of the following commands to force the update:

```
update.bat --force=true
update.sh --force true
```

Forcing an update by editing the setupDatabase.properties file

To force an update by editing the **setupDatabase.properties** file:

1. Open the **setupDatabase.properties** file in a text editor:

   Pega-image\scripts\setupDatabase.properties
2. Add the following line anywhere in the **setupDatabase.properties** file:

```
force.ml.update=true
```

3. Save and close the file.

4. At a command prompt, navigate to the *Pega-image*\scripts directory.
5. Run either **update.bat** or **update.sh**.

Appendix E — Secured mode

Deploying the Pega Platform in secured mode requires an administrator to enable new out-of-the-box operator IDs. The administrator and the following new out-of-the-box operators must change their passwords when they first log in:

- Batch@pega.com
- DatabaseAdmin@pega.com
- ExternalInviteUser
- IntSampleUser
- PRPC_SOAPOper
- PortalUser@pega.com
- UVUser@pega.com
- External

If you disable secured mode, the system creates and enables all new operators with default passwords. Unauthorized users might use the default passwords to gain administrator access to your system.

Note: When you deploy the Pega Platform in secure mode there are no changes to any existing administrator@pega.com operator in the target database. However, if there is no administrator@pega.com operator record, the deployment creates the operator.

For more information, see Running the Deployment Tool.

For information about enabling operators, see Enabling operators.

Appendix F — Rolling restart and Apache Ignite client-server mode

You can update Pega Platform to use the Apache Ignite client-server clustering topology to separate the Pega Platform processes from cluster communication and distributed features. Clustering technology has separate resources and uses a different JVM than Pega Platform. For more information, see Apache Ignite client-server clustering topology.

To enable client-server mode, follow the rolling restart process with additional steps for enabling client-server mode. See Performing the rolling restart.

To switch back from client-server mode to embedded mode, follow the rolling restart process with additional steps for disabling client-server mode. See Performing the rolling restart.

Performing the rolling restart

You can use a rolling restart process to enable Apache Ignite client-server mode or to switch back to embedded mode from client-server mode. For more details, see Apache Ignite client-server clustering topology.

Perform a rolling restart to keep your system always available during the update. Remove nodes from the load balancer, shut them down, update, and start them again one by one. You do not add them back to the load balancer until you have updated half the nodes.

To perform a rolling restart, complete the following steps:

1. Prepare the database.
    a. Disable rule saving. For more information, see Disabling rule creation on the rules schema.
    b. Migrate the PegaRULES schema to a temporary schema.
    c. Update the new rules schema, for example, a framework or application update.
    d. Copy the new rule schema to the production database.
2. Optional: To switch to Apache Ignite client-server mode, start the stand-alone Apache Ignite servers before updating the nodes. For more information, see Deploying and starting the Apache

Ignite servers.

3. Update half of the nodes one by one.
   a. Configure the load balancer to disable a node.
      - Disabling the node does not allow new connections, but it allows existing users and services to complete work.
      - Quiescing a Pega Platform node that has not been disabled in the load balancer results in error conditions for users of that Pega Platform node, because new users cannot log in. The Pega Platform must be disabled in the load balancer so that new users are redirected to another active Pega Platform node.
   b. Quiesce the Pega Platform node, by using the Autonomic Event Services (AES), System Management Administrator (SMA), or high availability landing pages. For more information, see the help for Cluster management and quiescing.
   c. Ensure that all requestors are passivated and the system run state is set to "Quiesce Complete", by using the HA Cluster Management landing page.
   d. Shut down the node.
   e. Update the data source to connect to the updated schema (to reflect changes made in step 1). For more information, see Upgrading the data schema.
      - Optional: To enable Apache Ignite client-server mode, modify the **prconfig.xml** file to switch the Apache Ignite cluster protocol and force the node to start in client mode.
        - Edit the **prconfig.xml** file and add the following settings:
          - `<env name="cluster/clientserver/clientmode" value="true" />`
          - `<env name="identification/cluster/protocol" value="ignite" />`
      - Optional: To switch back to embedded mode from client-server mode, modify the **prconfig.xml** file and remove the settings that were added during the switch to client-server mode.

- Edit the **prconfig.xml** file and remove the following settings:
  - `<env name="cluster/clientserver/clientmode" value="true" />`
  - `<env name="identification/cluster/protocol" value="ignite" />`

Note: To specify the technology that you want to use in embedded mode, enter the appropriate setting for the cluster protocol (Hazelcast or Ignite) instead of removing it.

f. Start the node.

g. Perform any needed post-update activities and tests. For more information, see Post-update configuration.

4. After you update half of the nodes, disable the remaining non-updated nodes in the load balancer.
5. Add all the updated nodes, which you updated in step 3, back to the load balancer to start taking traffic.
6. Update the remaining half of the nodes (the non-updated nodes) one by one.
   a. Perform steps 3 b through 3 g.
   b. Add the node back to the load balancer to start taking traffic.
7. Optional: To switch back to embedded mode from client-server mode, shut down all stand-alone Apache Ignite servers after all the nodes are updated and no longer use the stand-alone Apache Ignite server cluster.

Deploying and starting the Apache Ignite servers

The Apache Ignite servers provide base clustering capabilities, including communication between nodes. You must have a minimum of three stand-alone Apache Ignite servers for one cluster.

Deploy and start the Apache Ignite servers before you deploy and start Pega Platform.

1. Make sure that the JAVA_HOME environment variable points to a valid Java installation directory (JRE or JDK).
2. Move the **prcluster_service.war** file, which is used to start the cluster service, to the webapps

directory of your Pega Platform distribution image. The file is located in the Archives directory.

3. Configure the clustering protocol as Apache Ignite for all server nodes by using one of the following methods. Hazelcast cluster protocol is the default configuration.
   - Create a Dynamic System Setting, which is the recommended approach for configuring the cluster protocol. Dynamic System Settings are used to set preferences on all nodes at once, but you have to restart each node for the setting to take effect.

     For more details, see the help for creating Dynamic System Settings or the PDN article *How to compose the key of a PRCONFIG dynamic system setting*. To define the Dynamic System Setting, enter the following values:

     - **Pega-Engine** in the **Owning Ruleset** field.
     - **prconfig/identification/cluster/protocol/default** in the **Setting purpose** field.
     - **ignite** in the **Value** field.
   - Edit the **prconfig.xml** file that is used by the servers.
     a. Extract the content of the **prcluster_service.war** file in the webapps directory.
     b. Add the following setting to the **prconfig.xml** file that is located in the webapps/prcluster_service/WEB-INF/classes/ directory:

     ```
     <env name="identification/cluster/protocol" value="ignite" />
     ```
   - Pass the following JVM argument to the application server:

     ```
     -DNodeSettings=identification/cluster/protocol=ignite
     ```
4. Start your application server. The cluster service starts automatically.
5. After a successful startup, you can review the topology snapshot in the **PegaRULES** log files. By default, the log files are generated in the ../work/Catalina/localhost/prcluster_service/ directory and are accessible only from a terminal window.
6. Optional: Set up ELK (Elasticsearch, Logstash, and Kibana) for a convenient way to access and analyze the log files. For more information about configuring ELK, see the *Configuring Elasticsearch, Logstash, and Kibana (ELK) for log management* article on the PDN.