Perceptive Connect Runtime

Installation and Setup Guide

Version: 2.2.x

Written by: Documentation Team, R&D

Date: October 2024

Hyland™

Documentation Notice

Table of Contents

Overview

Perceptive Connect Runtime is the connector hub that allows you to create, configure, and enable channels that map data and functionality between Perceptive Software products and your various business applications.

Perceptive Software offers a variety of connectors for use with Perceptive Connect, Intelligent Capture, PeopleSoft, SAP, and more. These connectors provide triggers and actions for use in customizing channels that meet your needs and bridge the gaps between your business applications. For information about available connectors, contact your Perceptive Software representative.

This guide outlines the installation and configuration procedures for Connect Runtime. It also includes basic instructions for creating channels.

Prerequisites

To use Connect Runtime, you must have access to a working installation of the following software.

- Oracle Java SE 11, Oracle Java SE 17, Oracle OpenJDK 11, Oracle OpenJDK 17, Oracle Java SE 8
- One of the following supported web browsers, Google Chrome, Mozilla Firefox, or Safari

Overview of the setup process

To set up Connect Runtime, complete the following sections in order.

1. Install Connect Runtime
2. Install connectors
3. Create and configure a channel

Install Connect Runtime

Download the Connect Runtime

To download Perceptive product installation files, complete the following steps.

1. Go to the Hyland Community site.
1. From the menu, click **Support** and then under **Software Downloads** select **Perceptive Downloads**.
2. Find and download the installer file corresponding to the version to be installed.

Notes

New and updated documentation and help topics are regularly published to the documentation website at docs.hyland.com.

If this is an upgrade, extract the file to the Connect Runtime directory you previously used. When the system prompts you to merge the **Install** directory, click **Yes**.

Install the Connect Runtime

To install Connect Runtime, complete the following steps.

1. Extract the zip file to the directory in which you want to install Connect Runtime.

2. Open a terminal window and change to the **Connect Runtime** directory.

   Note Prior to executing the install command, if you are using either Java 11+ or 1.8 JDK on Windows, you may create an environment variable called **JVMDLL** that has the path to your **jvm.dll**. The **.dll** path is typically **%JAVA_HOME%\bin\server\jvm.dll**. This variable allows Hyland to preconfigure PCR's **JVM**. If you chose not to set the environment variable or are unable to use **JVMDLL** as a variable, you must configure the **JVM** when directed below.

3. To install Connect Runtime, complete one of the following actions.

   - If you are using the default service name, type the following command and then press **ENTER**:

     ```
     java -jar connect-runtime-installer-VERSION.jar install
     ```

   - If you are using a customized service name, type the following command and then press **ENTER**:

     ```
     java -jar connect-runtime-installer-VERSION.jar install <customized-service-
     name>
     ```

     Where **<customized-service-name>** is the name you want to use to register the Connect Runtime service.

   Notes

   - Do not use spaces in the customized service name. Make note of this name to use when starting the Connect Runtime service monitor program.
   - On a Java 11+ JDK, you may see the message **Error: org.osgi.framework.BundleException: Could not create bundle object in the console and logs**. This error is a side effect of changes in the Java 11+ JDK and has no effect on the installation. Once you start PCR, the changes in the **config.properties** file correct this issue.
   - On a Java 1.8 JDK, you may see the message **Unable to resolve pif-jdk11-compatability-bundle in the console and logs**. This bundle's purpose is to provide PCR with Java 11+ JDK compatibility. For any older Java JDKs, this bundle is unnecessary and will not resolve.

4. On a Java 11+ JDK, or an open JDK on Windows, if you did not create a JVMDLL environment variable, then you must configure PCR's JVM location. For more information, refer to Appendix B.

5. On a Java 11+ JDK, open your **system.properties** file located in the **conf** folder in your **Connect Runtime directory**. Under the **Jdk11 Compatibility Settings** section, uncomment each **javax.xml.*** property by removing the **#** at the start of each line.

6. To start the service, type the following command and then press **ENTER**:

   ```
   java -jar connect-runtime-installer-VERSION.jar start
   ```

   Note All data is stored in **H2 database by default**. If you prefer you may configure Connect Runtime to use Oracle, PostgreSQL or MSSQL. For more information, refer to Configure Database settings to configure the database

Upgrade the Connect Runtime

You can upgrade any 2.x PCR to a newer version. At this time, you cannot directly upgrade from a 1.x PCR to 2.x. You must migrate your 1.x PCR as directed by Migrate to Connect Runtime 2.x. To upgrade the Connect Runtime, complete the following steps.

1. Open a terminal window and change to the **Connect Runtime** directory.
2. To upgrade Connect Runtime, complete one of the following actions:

   - If you are using the default service name, type the following command and then press **ENTER**:

```
java -jar connect-runtime-installer-VERSION.jar upgrade
```

- If you are using a customized service name, type the following command and then press **ENTER**:

  ```
  java -jar connect-runtime-installer-VERSION.jar upgrade customized-service-name
  ```

  Where **customized-service-name** is the name, you want to use to register the Connect Runtime service.

Notes

- Do not use spaces in the customized service name. Make note of this name to use when starting the Connect Runtime service monitor program.
- Before the upgrade, the system creates an automatic backup. Refer to Roll back upgrades to restore from the backup.
- On a Java 11+ JDK, you may see the message **Error: org.osgi.framework.BundleException: Could not create bundle object in the console and logs**. This error is a side effect of changes in the Java 11+ JDK and has no effect on the installation. Once you start PCR, the changes in the **config.properties** file correct this issue.
- On a Java 1.8 JDK, you may see the message **Unable to resolve pif-jdk11-compatability-bundle in the console and logs**. This bundle's purpose is to provide PCR with Java 11+ JDK compatibility. For any older Java JDKs, this bundle is unnecessary and will not resolve.

3. On a Java 11+ JDK, or an open JDK on Windows, if you did not create a JVMDLL environment variable, then you must configure PCR's JVM location. For more information, refer to Appendix B.
4. On Windows, if you had any Java Options configured on the Java tab in **[drive:]\{*Connect Runtime directory*}\bin\<ServiceName>w.exe**, you will need to reconfigure the Java Options. You can copy the Java Options from the backup that was created during the upgrade from here: **[drive:]\{*Connect Runtime directory*}\Backup\snapshot-############\jvmargs.properties.** When pasting into the Java Options field, ensure there is a carriage return at the end of the last option.

   Note If one of your Java Options contained a drive letter in a file path, you will need to remove the backslash (\) before the colon (:). For example, if the following entry is in the **jvmargs.properties**, `-Djavax.net.ssl.trustStore=C\:/Program Files/Connect Runtime/data/pcr-truststore.jks` you will need to change it to the following when pasting it in the **Java Options** field, `-Djavax.net.ssl.trustStore=C:/Program Files/Connect Runtime/data/pcr-truststore.jks`.
5. On a Java 11+ JDK, open your **system.properties** file located in the conf folder in your Connect Runtime directory . Under the **Jdk11 Compatibility Settings** section, uncomment each **javax.xml.*** property by removing the # at the start of each line.
6. Type the following command and then press **ENTER**:

   ```
   java -jar connect-runtime-installer-VERSION.jar start
   ```

   Notes

   - If you upgrade from PCR 1.0.10 to PCR 1.1.x, the **pie.mapping.path** and **org.osgi.service.http.port** properties move from the **JVM Options** section of the service wrapper to the **config.properties** file.
   - If you upgrade from PCR 1.2.x to 1.3.x, your existing channel mappings will be migrated to the database you defined during Configure Database settings. As of PCR 1.3, we no longer store channel mappings on the file system and any new mappings will be stored in the database itself. As part of the upgrade to 1.3.x, the **pie.mapping.path** and **pie.mapping.backup.path** properties will be removed from the **config.properties** file. They will still be present in the copy of **config.properties** created with the backup prior to upgrading PCR.

- If you upgrade from PCR 1.3.x to 1.4.x and you have changed the **felix.webconsole.manager.root** setting in your **config.properties** file, you will need to remove this setting.

Migrate to Connect Runtime 2.x

Your current connectors will not run on the Connect Runtime 2.x. Prior to migrating, download the newest, PCR 2.x supported version of each connector installed in your current Connect Runtime. Also, we recommend you back up your current Connect Runtime 1.x to a location outside its install directory.

1. Export all your channels. For more information, refer to Import and Export Channels.
2. Manually record all connector and logger group configurations.
3. Complete the following sub-steps to uninstall the 1.x Connect Runtime.
    1. Open a terminal window.
    2. Type the following command to change the Connect Runtime directory and then press **ENTER**.

       ```
       java -jar connect-runtime-installer-1.x-VERSION.jar uninstall
       ```
4. Install the Connect Runtime 2.x. For more information, refer to Install the Connect Runtime.
5. Install the upgraded connectors. For more information, refer to the Install Connectors.
6. Configure your Connectors and logger groups.

   Note The connectors' configurations may change during the PCR 1.x to 2.x updates. For more information, refer to your connectors' documentation and Create and Configure Logger Groups.
7. Import your channels.

   Note The channel's configurations may change during the PCR 1.x to 2.x updates. For more information, refer to your connectors' documentation.

Rename the Connect Runtime UI

To change your service's name, complete the following steps.

1. Open a terminal window and change to the Connect Runtime Directory.
2. To unregister your instance, type the following command and then press **ENTER**:

   ```
   java -jar connect-runtime-installer-VERSION.jar
   ```
3. Navigate to the **conf** directory and edit the **serviceName** file. Save the new name to the service.name property.
4. To register your instance, type the following command and then press **ENTER**:

   ```
   java -jar connect-runtime-installer-VERSION.jar register
   ```

About the Connect Runtime UI

The Connect Runtime UI encompasses the Felix Web Console as well as pages for installing connectors, creating and configuring channels, and viewing created channels. The dashboard for the UI is found at **http://*{Connect Runtime host name}*:*{port}***. For example, **http://localhost:80**.

The default username and password for the Connect Runtime UI are both **admin**. These are configured through the Felix Web Console. To change these defaults, refer to the Apache Felix website for more information.

Note The Connect Runtime service runs on port 80 by default. To configure the port number, such as for multiple instances of Connect Runtime on the same host machine, refer to the Configure the Connect Runtime settings section in Appendix B.

Install connectors

About connectors

A connector is a bundle or collection of bundles that you can install in Perceptive Connect Runtime to enable the triggers and actions they define. We distribute connectors as either a JAR file or as a ZIP or PCR archive that contains a collection of JAR files. Refer to the connector's install guide for specific installation instructions.

Install a connector

To install a connector, complete the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **Install a Connector**.
2. To upload connector JAR, ZIP, or PCR files, drag and drop the files in the area designated by the box on the **Upload new bundles** page.

   Note Complete, queued, and processing files display at the bottom of the page. After Connect Runtime processes an item, a summary of the processed files displays, as well as an overview describing how many channels were affected by the installation.
3. Click **Accept** to accept the installation or **Roll back** to undo the installation. You must accept or roll back the installation before PCR can process the next item.

   Notes

   - Only one user may install connectors into Perceptive Connect Runtime at any given time. After a user begins the install process, they acquire a lock that PCR releases after the last file installs or after 15 minutes of inactivity.

     Note If the user does not accept or roll back an installed file from the report screen, PCR automatically rolls it back when the 15-minute timer expires.
   - The bundles in the installation may not start if any of the bundles are fragments.
   - You cannot see a summary of affected channels in any of the following cases.
   - The installation contains bundles with unversioned package exports.
   - The installation contains downgrades.
   - The installation contains bundle fragments.

The bundle installation summary

The bundle installation summary contains three categories.

- **Installed** specifies a list of bundles installed from the file or archive uploaded by the user.
- **Failed** specifies a list of bundles that failed to install. This includes a message with the reason for the failure.
- **Skipped** specifies a list of files that PCR cannot install, such as non-JAR files.

The affected channel summary

The channel summary contains three items.

- **Fixed** specifies a count of channels that previously failed to validate that became valid after the installation.
- **Broken** specifies a count of channels that previously validated that become invalid after the installation.
- **Unknown** specifies a count of channels that PCR cannot verify as valid or invalid for the report. This is usually a byproduct of older channels with trigger parameters that aren't stored in the database as plain text.

Verify a connector installation

To verify a connector installation, complete the following steps.

1. On the **PCR Dashboard**, under **Troubleshoot**, click **List OSGi Bundles**.
2. On the **OSGi Bundles** page, there is a list of installed bundles. Verify that the connector bundles you installed are in the **Active** state. If they are not, in the **Action** column, click the bundle's **Start** button.

Configure a connector

After installing a connector, you can configure it via the **Configuration** page, under the **Manage** heading. Refer to the connector documentation for specific configuration instructions.

Web Services

This page provides a list of all the possible REST and Soap Endpoints that can be configured. The system displays a **Shield** icon next to the endpoints that have been configured for a trust validator.

- No icon indicates that an endpoint has not been configured to a trust validator.
- A **blue shield** icon with a lock indicates a trust validator has been configured by the connector providing the endpoint and will not be modifiable.
- A **green shield** icon indicates a trust validator has been configured for the endpoint, and that it is active
- A **red shield** icon indicates a trust validator has been configured for the endpoint; however, there is something wrong with the configuration. The endpoint should not be up, and you should visit the **TROUBLESHOOT** links to determine what may be wrong.

Trust validators

Trust validators are provided services that are designed to protect REST/SOAP endpoints by ensuring that the user of the endpoints have valid credentials for its given system.

Configure trust validation

To configure a trust validation, complete the following steps:

1. On the **PCR Dashboard**, under **Manage**, click **Web Services**.

2. If you are on the home page, under **Manage Card**, click on the link. From here, you can select the Endpoint that you would like to add a trust validator. You can also view a list of all the possible REST and Soap Endpoints you can configure.

Add a trust validator

1. Select the endpoint by clicking on the row.
2. Click **Edit Trust Validator** on the toolbar above.
3. Select from the drop down list the validator you wish to be used.
4. Click **Save**.

Remove a trust validator

1. Select the endpoint by clicking on the row.
2. Click **Remove Trust Validator** on the toolbar above.
3. Confirm that you wish to remove the validator by clicking **Yes**.

Create and configure a logger group

A logger group consists of a name, file name, debug level, and a list of bundles for the resulting data.

Create a logger group

A logger group consists of a name, file name, debug level, and a list of bundles for the resultant data. To create a logging group, complete the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **ManageLogger Groups**.
2. In the **PCR Toolbar**, click **Create Group**.
3. In the right pane, in the **Group Propertie**s section, fill out the following properties.
   - **Group Name** specifies the name of this logger group.
   - **Log File Name** specifies the file path and name for which this group will be logged at.
   - **Logger Level** specifies the level at which the configured bundles will capture log data. This is a drop-down list.
   - **Enabled** specifies when to set to **True**, this property enables the group. When set to **False**, this property disables the group.
4. In the right pane, in the **Group Bundles** section, choose the bundles that you wish to include in this logger group by selecting the check box.
5. Click **Save** to save the group or click **Cancel** to cancel the operation. The **Manage Group Bundles** page is displayed.

Configure a logger group

You can modify a logger group at any time. Make the necessary changes and click **Save** for the changes to be recognized in PCR.

1. On the **Modify Logger Group Configure** page, modify any of the inputs you wish to change.

2. Modify any of the inputs you wish to change.
3. Click **Save** to save your changes or click **Cancel** to cancel your changes.

Create and configure a channel

A channel consists of a trigger, an action, and output mapping for the resulting data. To create a channel, complete the following steps.

Create a channel

A channel consists of a trigger, an action, and output mapping for the resultant data. To create a channel, complete the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **Create a Channel**.
2. **Optional**. In the right pane, in the **Channel** section, enter a name and description for the channel.

   Note Entering a name and description for the channel may help you find the channel on the **List Channels** page.
3. In the **Trigger Information** section, under **Trigger**, select a trigger from the list.
4. Fill out the trigger information and click **Continue**.

Configure a channel

1. On the **Modify Channel Mapping** page, under **Actions**, select an action for the channel.

   Note If you do not want the channel to perform an action, select **No Action**. For example, if your channel moves data using only readers and writers, you will select this option.

   If you are modifying the mapping for an existing channel, you will be prompted to confirm the modification prior to making changes.
2. If necessary, select the **Input** or **Output** tab to modify the input or output mappings.
3. If necessary, click **Save inputs** or **Save outputs** to save the input or output mapping.
4. If necessary, click **Validate inputs** or **Validate outputs**.
5. Any validation errors display in a pane to the right of the mapping area.
6. Click **Enable channel**.

Import and export configured channels

Channels can be imported and exported by the user. This is a nice way of transferring channels from one PCR instance to another.

Export channels

There are two ways to export channels; first, you can export a channel from the channel list page by selecting the channel and clicking **Export**. Second, you can export one to many channels by completing the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **Export Channels**.

2. Check the box next to each channel you want to export.
3. Click **Download Selected Channels**.

Note When more than one file is selected for export the downloaded file will be named ***channelExports.zip***, and when only one file is selected, the downloaded file will be named the ***name of the channel*** selected.

Import channels

To install a connector, complete the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **Import Channels**.
2. To upload XML, ZIP, or PCR files drag and drop the files in the area designated by the box on the **Upload new Channels** pane.

All files processed are displayed at the bottom of the page in a list view. When a file is expanded from the files list view, the following information is available.

- A list of all the files processed per the file that was imported.

  Each file can be selected by clicking on the file name, when a file is selected, a report will appear on the right. A report may contain the following information based on results from the import.
- Channel information
  - ID
  - Name
  - Action
- General errors
- Trigger errors
- Input errors
- Output errors

  By Clicking the **expand** icon above the report the entire report will pop out so that it is easier to read.

View configured channels

View a list of channels and deleted channels

The List Channels and List Deleted Channels pages list basic information for all configured and deleted channels in PCR. These pages can be viewed as a grid or a list. To view these pages, complete the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **List Channels** or **List Deleted Channels**.
2. Click **Switch to Grid View** or **Switch to List View** to view summary or detailed information about the channel.

Grid view

In Grid view, selecting the row of a channel allows you to perform additional actions on the channel. These actions may include enabling/disabling, cloning, editing mapping, deleting, or restoring the channel.

Grid view displays the following information about each channel.

- **ID** specifies the Connect Runtime's ID for the channel.
- **Name** specifies the user-defined name for the channel. Legacy channels display **Unnamed Channel**.
- **Trigger Name** specifies the name of the trigger the channel is configured to use.
- **Action** specifies the name of the action associated with the channel.
- **Trigger Parameters** specifies the configuration values entered when initially configuring the channel.

  Note Parameters for channels created before PCR version 1.0.10 are not available until you execute or view the channel in the mapping configuration screen after upgrading. Refer to Create and configure a channel for more information.

Additional columns for List Channels

- **Status** specifies whether the channel is **enabled**, **disabled**, or **unsatisfied**. An **enabled** channel is valid and can be executed by PCR. A **disabled** channel is not ready for execution by PCR. A channel is **unsatisfied** if the associated trigger or action are not registered with PCR. You cannot enable a channel if its state is unsatisfied.
- **Toggle Status** specifies whether a satisfied channel can be enabled or disabled using the toggle button in the rightmost column.

Additional columns for List Deleted Channels

- **Date Deleted** specifies the date and time the channel was deleted. The action buttons for a selected channel are in the toolbar with the **Refresh List** and **Switch View** buttons. See the button descriptions below for more information.

List view

List view displays each channel in an expandable pane. When the view is collapsed, List view displays the following information about each configured channel.

- **Status**
- **ID**
- **Name**
- **Description** specifies the user-defined description for the channel. Legacy channels will display "No channel description".
- **Action**
- **Trigger**
- **Trigger Parameters**

The following list provides additional information for List Deleted Channels.

- **Date Deleted**
- **State When Deleted**

When expanding a channel on the **List Channels** view, the following information and actions are available.

- **Input and Output Mapping** display the mapping configuration for the channel. Each has an expand icon above it that, when clicked, will display the mapping in a full-page view.
- **Edit Mapping**
- **Export**
- **Clone**
- **Enable or Disable**
- **Delete**

When expanding a channel on the **List Deleted Channels** view, the following information and actions are available:

- **Input and Output Mapping** display the mapping configuration for the channel. Each has an expand icon above it that, when clicked, will display the mapping in a full-page view.
- **Restore Channel**

Channel action buttons

The following describes the Cancel action buttons.

- **Toggle Status** enables or disables a satisfied channel.
- **Edit Mapping** opens the channel’s modify mapping page if the selected channel is not enabled.
- **Export** downloads the channel to the file system. A channel in any state can be exported. The name of the exported file will be its channel name.
- **Clone** opens the create channel page and pre-fills some of the source channel's information if the selected channel is satisfied. Upon saving the new channel, the mapping page will contain a copy of the original channel's input and output mappings.
- **Delete** prompts you for confirmation and then deletes the selected channel.
- **Restore Channel** restores the selected channel. You cannot restore a channel if the channel would be a duplicate of an existing channel.

  Note Restore Channel is only available for deleted channels. Toggle Status, Edit Mapping, Clone, and Delete are only available for non-deleted channels.

Manage logger groups

Manage a list of configured logger groups

The Manage Logger Groups page list basic information for all configured logger groups as well as provides a way to create and modify logger groups in PCR. You can view these pages as a grid or a list. To view these pages, complete the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **Manage Logger Groups**.
2. Click **Switch to Grid View** or **Switch to List View** to view summary or detailed information about the logger group.

Grid view

In Grid view, selecting the row of a logger allows you to perform additional actions on the logger group. These actions may include enabling/disabling or editing the logger group. Grid view displays the following information about each logger group.

- **Group Name** specifies the user-defined name for the logger group.
- **File Name** specifies the name of the file where the logged data is written.
- **Level** specifies the level at which the debug is set.
- **Status** specifies whether the logger group is enabled or disabled.

An enabled group will provide logging to the designated file.

A disabled group will not provide logging to the designated file Additional columns for Logger Groups

- **Group Bundles** specifies the list of bundles that are configured to log messages at the configured level of the logger group.

Additional columns for Group Bundles

- **Bundle Name** specifies the bundle name(pretty name) of the bundle that can be configured for the logger group.
- **Symbolic Name** specifies the is the symbolic name of bundles that are already configured or can be configured.
- **Status** specifies whether the configured bundle is missing or installed in PCR.

You can filter the Group Bundles by typing in in the filter input box.

The action buttons for a selected logger group are in the toolbar with the **Refresh List** and **Switch View**

Buttons. See the button descriptions below for more information.

List view

List view displays each logger group in an expandable pane. When the view is collapsed, List view displays the following information about each configured logger group.

- **ID** specifies the connect Runtime's ID for the logger group.
- **Group Name**
- **File Name**
- **Level**

When expanding a logger group on the **Manage Logger Groups** view, the following information and actions are available.

- **Status** specifies the status of the logger group, whether it is enabled or disabled.
- **Group Bundles** displays the selected bundles for the logger group. There is an expand icon above it that, when clicked, will display all available bundles in a full-page view.
- **Edit Group**
- **Enable** or **Disable**
- **Delete**

Logger group action buttons

The following describes the Logger group action buttons.

- **Toggle Status** enables or disables a logger group.
- **Edit Group** opens the logger group's modify group page.
- **Create Group** opens the logger group's create group page.
- **Delete** prompts you for confirmation and then deletes the selected logger group.

View logs

View log messages generated by the system

The View Logs page lists messages logged by the system. Along with the message, this page displays the received time, log level, message source, and message exception. Only the last 499 records in a log group are displayed, however, if your group contains more records, then this page can be viewed as a grid or a list. To view this page, complete the following steps.

1. On the **PCR Dashboard**, under **Troubleshoot**, click **View Logs**.
2. Click **Switch to Grid View** or **Switch to List View** to view summary or detailed information about the log messages.

Grid view

Grid view displays the following information about each log message.

- **Received** specifies the time the message was written.
- **Level** specifies the level at which the debug is set.
- **Source** specifies the name of the Bundle, Service, and Class the debug originated from.
- **Message** specifies the message.
- **Exception** specifies the Exception message, and stack trace based on filter.

List view

List view displays the following information about each log message.

- **Received** specifies the time the message is written.
- **Level** specifies the level at which the debug is set.
- **Message** specifies the message.
- **Exception** specifies the Exception message, and stack trace based on filter.
- **Logger Source** specifies the name of the class the debug originated from.
- **Bundle** specifies the name of the bundle when known.
- **Service** specifies the name of the service and details of the service if applicable.

View filters view

You can take the following actions to filter your log messages.

- **Group Identifiers** specifies all the log groups previously created, as well as the "Global Log" which contains all the log messages. Choose the group for which you are wanting to see log messages.
- **Minimum Log Level** specifies all the available log levels that a group can be logging at. Choose the minimum level to which you want to see log messages
- **Exception Details** specifies the exception details. The options are **Message Only** and **Full Stack Trace**.
- **Message Only** displays only the exception message of the exception being logged.
- **Full Stack Trace** displays the full stack trace of the exception being logged.
- **Filter By Date**: filters the log messages from the received time. Valid operators are: **Operators** =,>,>=,<,<=
- **Filter Bar** searches and displays all matched. The filter bar filters from the entire log message list.

Appendix A: Upgrade connectors

Create a backup

Before upgrading a connector, we recommend that you back up your current PCR installation. This saves a snapshot of your current connector configurations that you can revert to if you encounter upgrade issues.

To create a backup of PCR, complete the following steps.

1. Open a terminal window and change to the **Connect Runtime directory**.
2. Type the following command and then press **ENTER**:

```
java –jar connect-runtime-installer-VERSION.jar backup
```

Note The Connect Runtime installer creates a folder called Backup in **Connect Runtime directory** where it stores the backup snapshot you created.

Upgrade a connector

To upgrade an installed connector, complete the process outlined in "Install a connector" and select the updated connector files to install. When you update the bundles, the PCR overwrites the existing connector with the newconnector.

Roll back upgrades

If you need to undo a connector upgrade and return to a previous installation and configuration of Connect Runtime, complete the following steps.

1. Open a terminal window and change to the **Connect Runtime directory**.
2. Type the following command and then press **ENTER**:

```
java –jar connect-runtime-installer-VERSION.jar rollback
```

3. Type the numeral of the backup snapshot that you want to rollback to, shown in the list, and then press **ENTER**.

Appendix B: Customize and configure Connect Runtime

Use the methods in the following sections to troubleshoot Connect Runtime.

Configure Connect Runtime logging

You use the Connect Runtime configuration page in the UI to configure the logging level and log directory. To access the console logging, complete the following steps.

1. From any page within the runtime UI, click **Configuration** under the **Manage** heading.
2. In the **Name** column, under **General**, click **PIF Logger**.
3. Set the **Log Level** and **Log Directory**.
4. Optional. Select either of the**Rollover Policies**.
5. If **Time** is selected, set the **Time-Based Rollover Interval**.
6. If **File size** is selected, set the **Maximum File Size** (MB).
7. If either are selected, set the **Archive Size**.
8. Click **Save**.

Note You do not need to restart Connect Runtime to implement the settings.

Verify Perceptive Connect Runtime service is running in Linux

You can run the following Linux command to check the status of Connect Runtime Service.

```
systemctl status<servicename>
```

Where **<servicename>** is **perceptiveconnectruntime** by default, otherwise it is the customized service name you used when installing or upgrading Connect Runtime.

Run the Connect Runtime service monitor program in Windows

You use the Connect Runtime service monitor program to view PCR properties. You can view the properties in each tab of the program menu. To implement changes, restart PCR.

To start the monitor program, complete the following steps.

1. Navigate to the **[drive:]\{*Connect Runtime directory*}** directory.
2. Run **<ServiceName>w.exe**, where **ServiceName** is **PerceptiveConnectRuntime** by default, otherwise it is the customized service name you used when installing or upgrading Connect Runtime.

Connect Runtime properties

The PCR service monitor program provides the following information.

- **General** specifies the install path and the Startup type. Shows whether the service is running. You can stop or start the service from this tab.
- **Log On** specifies the account settings.
- **Logging** specifies the logging level, the log path, and the log prefix. You can change the location of the log file from this tab. If specified, the log directory setting in the Connect Runtime Web Console overwrites this log path.

- **Java** specifies the path of the Java Virtual Machine, the path of the Java Classpath, the port, memory settings, and stack size. To change the port setting, refer to Configure the Connect Runtime settings. The memory settings and stack size should remain blank. To change the JVM settings, uncheck **Use default** above Java Virtual Machine and do one of the following:
  - Paste the path of your JDK’s **jvm.dll** file into the **text** box.
  - Click the **ellipsis** button and navigate to you JDK’s jvm.dll file, select it and then click **Open**.
- **Startup** specifies the Connect Runtime working path. This is optional. Other settings in this box should remain blank.
- **Shutdown** settings in this box should remain blank. This is optional.

Note If you get the error message "The specified service does not exist as an installed service" when launching the Connect Runtime service monitor, complete the steps in the Register the Connect Runtime service section.

Configure the Connect Runtime settings

To open the Connect Runtime configuration file, complete the following step.

- Navigate to the **Connect Runtime directory/conf** directory and open the **config.properties** file in a text editor.

Configure the port settings

The Connect Runtime service runs on port 80 by default. However, each instance of Connect Runtime on a single host machine must have a unique port number. To configure a Connect Runtime instance port number, complete the following steps.

1. In the **config.properties** file, set **org.osgi.service.http.port** to your port number. For example, if the port is 7000, the setting is **org.osgi.service.http.port=7000**
2. Save the file and restart the runtime.

Configure database settings

Perceptive Connect Runtime stores channel mappings into the database defined by these settings. In the case that these database settings are omitted, any PCR data is persisted in the in-memory H2 database provided through PCR.

Perceptive Connect Runtime supports the following databases **H2**, **MSSQL**, **PostgreSQL** and **Oracle**.

Add the following properties to the **config.properties** file.

- **pcr.db.server.type=[serverType]**
- **pcr.db.servername=[serverName]**
- **pcr.db.port=[portNumber]**
- **pcr.db.databasename=[databaseName]**
- **pcr.db.username=[userName]**
- **pcr.db.password=[password]**

Where **serverType** is the type of database PCR connects to the choices are **h2, mssql**, and **oracle**. **"postgresql"serverName** is the IP address, or DNS name of your database server; **portNumber** is the open port that your database is listening on; *** databaseName** is the name of the database that PCR will use to store all of its data; **userName** is the database user with access to the database **password** is the database required password.

Example:

```
pcr.db.server.type=mssql
pcr.db.servername=PCR_DB_SERVER
pcr.db.port=1433
pcr.db.databasename=PCR_DATA
pcr.db.username=pcrUser
pcr.db.password=imagenow
```

Notes By default when Perceptive Connect Runtime is installed, H2 is the chosen database, and these properties are not required. For MSSQL, you must create a PostgreSQL or Oracle Databases a database to store the PCR data. You can name this anything. For MSSQL, PostgreSQL or Oracle Databases a user must have privileges to update, create and delete tables in the PCR database.

Configure the session timeout

The default timeout for the Connect Runtime is 20 minutes. To change this time frame, complete the following steps.

1. In the **config.properties** file, change the value of **org.apache.felix.http.session.timeout** to a duration in minutes, such as **org.apache.felix.http.session.timeout=60**.
2. Save the file and restart the runtime.

Additional configure for Linux

To open the Connect Runtime **linuxEnvironmentFile** configuration file, complete the following step.

To customize java location

1. Navigate to the **Connect Runtime directory/conf** directory and open the **linuxEnvironmentFile** file in a text editor.
2. Set **PCR_JAVA_BIN** to the path to which java bin you would like to run. By default, the java used will be the java found in your Linux path. For example, **PCR_JAVA_ BIN='/bin/java'**.
3. Save the file and restart the service.

To add any additional JVM arguments

1. In the **linuxEnvironmentFile** file, set **PCR_JVM** to include any JVM arguments you wish to pass to Connect Runtime. For example, **PCR_JVM='-Xdebug -Xrunjdwp:transport=dt_ socket,server=y,suspend=n,address=1044'.**
2. Save the file and restart the service.

About configuration settings with file paths

Java Properties uses the **config.properties** file. As a result, all settings that contain file paths should use forward slashes ('/') and not backslashes ('\'). For more information, refer to Java Properties documentation on the Oracle website.

Configure the runtime to allow incoming CORS requests

Note If you are using a reverse proxy server, firewall, or load balancer to allow communication between PCR and your web application), you may not need to enable CORS.

If your installation of Connect Runtime will be receiving AJAX requests from an external web application, you must enable CORS. CORS, simply put, is what allows Javascript code running on one domain to access resources on another domain.

To enable CORS in PCR, make the following changes to the **config.properties** file:

```
connect.rs.cors.enabled=false
connect.rs.cors.allow.credentials=false
connect.rs.cors.allow.all.origins=false
connect.rs.cors.allowed.origins=http://domain1.com, http://domain2.com
```

- **connect.rs.cors.enabled** specifies whether to enable CORS in PCR.
- **connect.rs.cors.allow.credentials** specifies whether your web application needs to pass secure data, such as cookies or authentication headers, to PCR.
- **connect.rs.cors.allow.all.origins** specifies whether to allow CORS requests from any external application.
- **connect.rs.cors.allowed.origins** specifies a list of domains from which CORS requests are permitted. Each domain must be in the format **http(s)://domain.com:port**. CORS requests coming from domains not in this list will be denied.

Notes

- The values of **connect.rs.cors.allow.all.origins** and **connect.rs.cors.allow.credentials** cannot both be set to **true**.
- The value of **connect.rs.cors.allowed.origins** is ignored if **connect.rs.cors.allow.all.origins** is set to **true**.

Configure PCR for SSL

Modify the following properties in the **config.properties** file.

```
org.apache.felix.https.enable=true
org.osgi.service.http.port.secure=443
org.apache.felix.https.keystore=[full path to keystore.jks file]
org.apache.felix.https.keystore.password=[password]
org.apache.felix.https.keystore.key.password=[password]
org.apache.felix.https.truststore=[full path to the truststore.jks file]
org.apache.felix.https.truststore.password=[password]
```

A description of each setting is as follows:

- **org.apache.felix.https.enable** enables SSL in the runtime.
- **org.osgi.service.http.port.secure** specifies the port used to accept SSL requests.
- **org.apache.felix.https.keystore** specifies the path for the keystore created in **[Create a keystore for PCR].**
- **org.apache.felix.https.keystore.password** specifies the password for the keystore.
- **org.apache.felix.https.keystore.key.password** specifies the password for the key (alias) in the keystore.

Note By default this is the same as org.apache.felix.https.keystore.password, but that is not a requirement.

- **org.apache.felix.https.truststore** specifies the path for the truststore created in [Create a truststore for PCR].
- **org.apache.felix.https.truststore.password** specifies the password for the truststore.

Note For further SSL configuration information, refer to the documentation on the Apache Felix website.

Uninstall Connect Runtime

The uninstall action removes the Connect Runtime service from Windows services. It does not delete the installation directory. To uninstall Connect Runtime, complete the following steps.

1. Open a terminal window and change to the **Connect Runtime directory**.
2. Type the following command and then press **ENTER**:

```
java –jar connect-runtime-installer-VERSION.jar uninstall
```

Register the Connect Runtime service

The installer automatically registers the Connect Runtime Service with Windows when installing. However, it may be necessary to re-register the service, particularly when moving the Connect Runtime to another location on disk. To re-register the service, complete the following steps.

1. Open a terminal window and change to the **Connect Runtime directory**.
2. Type the following command and then press **ENTER**:

```
java –jar connect-runtime-installer-VERSION.jar register
```

Unregister the Connect Runtime Service

You can unregister the Connect Runtime service without uninstalling Connect Runtime. To unregister the service, complete the following steps.

1. Open a terminal window and change to the **Connect Runtime directory**.
2. Type the following command and then press **ENTER**:

```
java –jar connect-runtime-installer-VERSION.jar unregister
```

Appendix C: Actions

An action is a connector-defined task configured in the channel. The action executes when the channel is triggered.

Perceptive Connect Runtime provides the ability, by default, to select **No Action** as an option for any connector you install on Connect Runtime. This interface is available for use in connector channels where you only use readers and writers. It does not perform any logic.

For more information about other channel actions, refer to the installation guide for any Perceptive connector.

Appendix D: Readers

Readers are components that let you configure connector channels to retrieve values from other applications. The channel uses these values in the data context when it runs. You use readers to configure the action input mapping. Input mappings allow a channel to have the required data to execute the action. You can invoke readers using specific XML tags, which Perceptive Connect defines per reader.

Some readers, such as the Date transformer and the XML transformer, take a set of data and transform it.

Note The channel data context refers to the data from various sources that is available in the channel for action input and results output configuration. The data available in the context depends on the trigger the channel uses as well as the connectors that are installed in Connect Runtime.

Perceptive Connect Runtime provides the following readers for use with any connector.

Date Format reader

The Date Format reader is used to transform date strings between formats, such as mm/dd/yyyy and yyyy-dd-mm. The reader takes a date string from the parameter reference, transforms the format, and outputs the resulting date string. The Date Format reader provides the following configuration fields.

- **dateReference** specifies a mapped string parameter that represents a date.
- **inputFormat** specifies the **dateReference** (input) date format.
- **outputFormat** specifies the output date format.

To use the Date Format reader, use the following XML format within a parameter.

```
<c:dateFormatter>
      <c:dateReference></c:dateReference>
      <c:inputFormat></c:inputFormat>
      <c:outputFormat><c:outputFormat/>
</c:dateFormatter>
```

The date format strings should follow the patterns used by the Java **SimpleDateFormat** class. You can find more information on the Oracle website.

Example

```
<c:parameter>
  <c:name>dateInput</c:name>
  <c:literal>2014-08-12</c:literal>
</c:parameter>
<c:parameter>
  <c:name>formattedDate</c:name>
  <c:dateFormatter>
    <c:dateReference>dateInput</c:dateReference>
    <c:inputFormat>yyyy-dd-mm</c:inputFormat>
    <c:outputFormat>mm/dd/yyyy</c:outputFormat>
  </c:dateFormatter>
</c:parameter>
```

In this example, the Date Format reader transforms the **dateInput** value of **2014-08-12** to an output of **12/08/2014**.

List Item reader

The List Item reader lets you access a single value from a list of items provided by a trigger or a reader.

To use the reader, include the following XML template in the channel input mapping, complete with the appropriate values.

```
<parameter>
  <name></name>
    <listItem>
      <listRef></listRef>
      <index></index>
    </listItem>
</parameter>
```

The **listRef** field contains a reference to the list item you want to access. The **index** field specifies the location of the list item in the index. Index numbers start at “0."

Positive Index

```
<parameter>
  <name>myItem</name>
  <listItem>
    <listRef>myList</listRef>
    <index>3</index>
  </listItem>
</parameter>
<c:parameter>
  <c:name>myList</c:name>
  <c:trigger>Results</c:trigger>
</c:parameter>
```

In the example, the reader retrieves the item at the specified index number, from the list **listRef** associated with the **index** number, and stores it in the context as **myItem**.

Negative Index

```
<parameter>
  <name>myItem</name>
    <listItem>
      <listRef>myList</listRef>
      <index>-2</index>
    </listItem>
</parameter>
<c:parameter>
  <c:name>myList</c:name>
  <c:trigger>Results</c:trigger>
</c:parameter>
```

In the example, the reader retrieves the item at the specified index number, from the list **listRef** associated with the **index**number. The negative index number tells the reader to retrieve the second-to-last element from the list and stores it in the context as **myItem**.

Literal reader

The Literal reader reads a literal string and then stores the value in the data context for further use. To use the Literal reader, enter the following XML within a parameter.

```
<c:literal></c:literal>
```

Example

```
<c:parameter>
  <c:name>eFormName</c:name>
  <c:literal>AP Invoice</c:literal>
</c:parameter>
```

In this example, the Literal reader reads and stores the value **AP Invoice** under the name **eFormName**. You can reference **eFormName** later in the data context, as needed, to retrieve the value **AP Invoice**.

Regular Expression transformer

The Regular Expression transformer completes a RegEx find and replace operation on a string. It provides the following configuration fields.

- **find** specifies the string or regular expression it searches for.
- **replace** specifies the string or regular expression to replace the matched text.

You should format both fields in a manner that the Java Pattern class's **compile()** method can accept.

To use the regular expression transformer, use the following XML format.

```
<c:regexTransform>
  <c:find></c:find>
  <c:replace></c:replace>
</c:regexTransform>
```

Example

```
<c:parameter>
  <c:name>newDepartmentName</c:name>
    <c:regexTransform>
      <c:reference>departmentName</c:reference>
      <c:find>R&amp;D</c:find>
      <c:replace>AP/ERP</c:replace>
    </c:regexTransform>
</c:parameter>
```

In this example, the Regular Expression transformer finds the **departmentName** item in the data context and then searches for the string "R&D". Then, it replaces and all occurrences with the phrase "AP/ERP". The transformer stores the result in the **newDepartmentName** context item for use during the channel's execution.

Example

```
<c:parameter>
  <c:name>trimmedText</c:name>
    <c:regexTransform>
      <c:reference>rawText</c:reference>
      <c:find>^\s*(.*?)\s*$</c:find>
      <c:replace>$1</c:replace>
    </c:regexTransform>
</c:parameter>
```

In this example, the Regular Expression transformer removes any leading or trailing whitespace from the **rawText** data context. The transformer stores the result in the **trimmedText** context item for use during the channel's execution.

Stream to XML reader

The **StreamToXML** reader takes a reference to a **MappingInputStream** object and transforms it into an **org.w3c.dom.Document** object.

To transform a **MappingInputStream** into an **org.w3c.dom.Document** object, include the following XML template in the channel input mapping, complete with the appropriate values.

```
<c:streamToXML>
```

```
  <c:streamRef></c:streamRef>
</c:streamToXML>
```

The **streamRef** field is a reference to the **MappingInputStream**.

Example

```
<c:parameter>
  <c:name>xmlDoc</c:name>
  <c:streamToXML>
    <c:streamRef>file</c:streamRef>
  </c:streamToXML>
</c:parameter>
<c:parameter>
  <c:name>file</c:name>
  <c:trigger>FileStreamParam</c:trigger>
</c:parameter>
```

In the example, **trigger** provides a **MappingInputStream** parameter called **FileStreamParam**. **FileStreamParam** is stored in the context as **file**, which is passed into the **streamToXML** reader. The parsed XML Document is then stored in the context as **xmlDoc**. The XML reader or XML transformer can then use the parameter.

Trigger reader

The Trigger reader reads values in the data context that are provided as outputs by the channel's trigger. To use the Trigger reader, enter the following XML.

```
<c:trigger></c:trigger>
```

For example, the Perceptive Intelligent Capture Connector provides the Export trigger. This trigger provides the Content document ID as an output called **DocumentId**. To read that output value and store it in the data context for further use with the reference **DocId**, enter the following XML.

```
<c:parameter>
  <c:name>DocId</c:name>
  <c:trigger>DocumentId</c:trigger>
</c:parameter>
```

XML reader

The XML reader reads values from an XML document. The reader processes repeating elements and single elements. The XML reader provides the following configuration fields.

- **reference** specifies the data context value that contains the XML document. If the **xmlRowSource** node is not nested under another **xmlRowSource** node, you must use this field. Otherwise, it is optional.
- **context** specifies the parent element. If the **xmlRowSource** node is nested inside another **xmlRowSource** node, you must use this field. Otherwise, it is optional.

  Note You must use either the reference or context field in each **<c:xmlSource>** or **<c:xmlRowSource>** instance.
- **id** sets a unique value for **xmlSource** and **xmlRowSource** children to use when specifying their context element.
- **xpath** specifies the **Xpath** location, in the XML document, of the values to read.

  Note An XPath element can also contain a number or string literal of the form , such as number("1") or string ("mystring"), respectively.

To read repeating elements, you would use an **xmlRowSource** node and the **reference** or **context, id**, and **xpath** fields. Use the following XML format.

```
<c:xmlRowSource>
  <c:reference></c:reference> OR <c:context></c:context>
  <c:id></c:id>
  <c:xpath></c:xpath>
</c:xmlRowSource>
```

To read a single element from an XML document, you use an **xmlSource** node and the **reference** or **context** and **xpath** fields. Use the following XML format.

```
<c:xmlSource>
  <c:reference></c:reference> OR <c:context></c:context>
  <c:xpath></c:xpath>
</c:xmlSource>
```

The following examples assume there is a trigger that provides an XML document. The Trigger reader names the provided document **XMLDoc** in the data context. Then this component reads the value at **IntelligentCaptureDocument/InvHeader/INVOICE_Number** from the referenced XML document.

Note Perceptive Connect Runtime uses **Java XPathExpression** to evaluate xpath. You must provide a namespace, as shown in the examples below, to search and map additional items.

Single element

```
<c:parameter>
  <c:name>XMLDoc</c:name>
  <c:trigger>XmlDocument</c:trigger>
</c:parameter>
<c:parameter>
  <c:name>InvNum</c:name>
  <c:xmlSource>
    <c:reference>XMLDoc</c:reference>
    <c:xpath>/*[local-name()='transcript' and namespace-
uri()='http://www.hyland.com/transcript/xml/bwout']/type</c:xpath>
  </c:xmlSource>
</c:parameter>
```

Repeating elements

```
<c:parameter>
  <c:name>XMLDoc</c:name>
  <c:trigger>XmlDocument</c:trigger>
</c:parameter>
<c:rowset>
  <c:name>page</c:name>
  <c:xmlRowSource>
    <c:reference>XMLDoc</c:reference>
    <c:id>Root</c:id>
    <c:xpath>/IntelligentCaptureDocument</c:xpath>
  </c:xmlRowSource>
  <c:mapping>
    <c:parameter>
      <c:name>InvNumber</c:name>
      <c:xmlSource>
        <c:context>Root</c:context>
        <c:xpath>InvHeader/INVOICE_NUMBER</c:xpath>
      </c:xmlSource>
    </c:parameter>
  </c:mapping>
</c:rowset>
```

XML transformer

The XML transformer takes a set of data and transforms it into a provided XML template or schema. The XML transformer provides the following three configuration fields.

- **data** is a reference to a ContextList in the data context. The value used in the data field needs to be a rowset of values from the XML reader.
- **template** specifies the XML document into which the XML transformer writes and saves the data. Any repeating elements in the template only need to have one instance or the exact number of instances found in data. Otherwise, the component will not work.
- **mapping** specifies list of key and value pairs that override the transformation location of some values in the data element.

To use the XML transformer, use the following XML format.

```
<c:xmlTransform>
  <c:data></c:data>
  <c:template></c:template>
  <c:mapping>
    <c:entry>
      <c:key></c:key>
      <c:value></c:value>
    </c:entry>
    ...
    </c:mapping>
</c:xmlTransform>
```

For a complete example of how to use the XML transformer in your channel, refer to the XML example provided with your connector.

Null Reader

The NullReader maps the Java null reference into parameters required by Actions. Channels should only use the NullReader if the Action has logic specifically to deal with ExplictNull values.

To use the Null Reader, use the following XML format.

```
<c:parameter>
  <c:name>ActionParameterThatCanBeNull</c:name>
  <c:null/>
</c:parameter>
```

Appendix E: Configure SSL

Perceptive Connect Runtime has full support for SSL/TLS, including client-side authentication. Use the following methods to configure SSL.

About Inbound and Outbound SSL/TLS configuration

Perceptive Connect Runtime can accept as well as initiate network communications. Therefore, you may need to configure the system for both inbound and outbound SSL/TLS.

About the config.properties file

The **config.properties** file contains most of the configuration settings for PCR. You can find it in the [connect install directory]/conf/ directory. Note that when modifying settings in this file, you cannot have any trailing spaces.

Note There are some parameters that need to be configured by setting JVM options before PCR starts. For further information, refer to Configure outbound SSL connections.

Configure inbound SSL connections

When a machine initiates a connection to PCR, it is considered an inbound connection. Typical examples of inbound connections to PCR are from an administrator's web browser to configure PCR, from external systems that integrate with PCR like Perceptive Content's Workflow, or from other applications making data lookup calls into REST services provided by Connectors installed in PCR.

You must configure SSL/TLS to provide secure PCR connections. At a minimum, you are required to install a signed server certificate into a keystore.

To configure inbound SSL connections, complete the following steps.

1. In the **[connect install directory]/conf/** directory, open the **config.properties** file with a text editor.
2. Add the inbound SSL connections settings, replacing **[connect install directory]** and **[password]** with the path and password used in your keystore. You can configure the **org.osgi.service.http.port.secure** setting to any valid port.
3. Save and close **config.properties** file.
4. Optional. To test the configuration, use a browser to connect to the **PCR Dashboard** on the SSL port you configure. For example, **https://myserver:443/**.

Example

```
org.apache.felix.https.enable=true
org.osgi.service.http.port.secure=443
org.apache.felix.https.keystore=[connect install directory]/data/pcr- keystore.jks
org.apache.felix.https.keystore.password=[password]
```

Advanced inbound SSL configuration

Perceptive Connect Runtime supports the following inbound SSL configuration in the **config.properties** file. For more information, refer to the Apache Felix HTTP Service documentation.

Setting	Options	Description
org.apache.felix.https.enable	TRUE FALSE	Enables SSL in the runtime.
org.osgi.service.http.port.secure	Any valid port number	Specifies the port used to accept SSL requests.
org.apache.felix.https.keystore	Any valid path	Specifies the path for the PCR keystore. PCR may have its own keystore or share a keystore with other applications on the system.

org.apache.felix.https.keystore.password		Specifies the password for the keystore.
org.apache.felix.https.keystore.key.password		Specifies the password for the key (alias) in the keystore. **Note** By default, this is the same as **org.apache.felix.https.keystore.password**, but the two passwords do not have to match.
org.apache.felix.https.truststore		Specifies the path for the PCR truststore. PCR may have its own truststore or share a truststore with other applications on the system. For example, you might use the default Java truststore in the **[JAVA_HOME]/lib/security/cacerts** file. You can find more information on the Oracle website.
org.apache.felix.https.truststore.password		Specifies the password for the truststore.
org.apache.felix.https.clientcertificate		Specifies the client authentication on inbound connections. Legal values are needs, wants and none. The default is none.

Configure outbound SSL connections

When PCR initiates a connection to another machine, it is considered an outbound connection.

For outbound connections, PCR uses the Java Secure Socket Extension (JSSE) truststore. By default, the JSSE trust store is in the Java install directory. If you need to modify the default JSSE truststore, it is recommended that you configure the Connect Runtime to use the truststore you specified for the inbound SSL connections. To configure the JSSE truststore, complete the following steps.

Note In many cases PCR will be initiating SSL/TLS connections to servers that have certificates signed by a Certificate Authority that is trusted by the default JSSE truststore. In that case, you will not need to complete these steps. However, if PCR needs to initiate a connection to a server with a self-signed certificate, you may need to complete the following steps.

Windows Steps

1. Navigate to **[connect install directory]/bin/**.
2. Run **PerceptiveConnectRuntimew.exe**
3. On the **Java** tab, in the **Java Options** section, add the setting **-Djavax.net.ssl.trustStore= [*connect install directory*]/data/pcr-truststore.jks**. Specify the file configured for the **org.apache.felix.https.truststore** setting in the **config.properties** file.

   Note Replace [connect install directory] with the path to the connect installation.

Linux steps

1. Navigate to the **[Connect Runtime directory]/conf** directory and open the **linuxEnvironmentFile** file in a text editor.

2. Add or modify **PCR_JVM** by setting **PCR_JVM='-Djavax.net.ssl.trustStore=[*connect install directory*]/data/pcr-truststore.jks'**. Specify the file configured for the **org.apache.felix.https.truststoresetting** in the **config.properties** file.

   Note Replace [connect install directory] with the path to the connect installation.

Configure client authentication

To enable client validation, complete the following steps.

1. Navigate to **[connect install directory]/conf/**.
2. Open the **config.properties** file with a text editor.
3. Add **org.apache.felix.https.clientcertificate=needs**.
4. Configure clients to provide appropriate certificates.

Notes The exact steps required depend on the client. Most browsers are either configured in the OS or have their own configuration for client certificates. Other web services have their own configuration. **-** If a client attempts to connect without a valid certificate, an error such as the following occurs in the pif.all.log file.

```
2014-11-06 10:30:19 [o.e.jetty.io.nio] WARN    -
javax.net.ssl.SSLProtocolException: handshake alert: no_certificate
```

Appendix F: Metrics

PIF Metrics allow you to capture performance data for the framework as well as individual connectors. Included are several built-in categories for measuring metrics over varying amounts of time. PIF Metrics writes output data as a CSV file in a user-defined location.

Configure Metrics

By default, Connect continuously collects and writes metrics to a file over a four-hour period. To configure metrics, complete the following steps.

1. On the **PCR Dashboard**, under **Manage**, click **Configuration**.
2. In the right pane, in **Name** column, under **General**, click **PIF Metric**s.

Metrics settings

Connect Metrics include the following settings. **Metrics directory**: The path to the directory for writing Connect Metrics. **Metrics Categories**: The Metric categories to report. **Continuous Collection**: If enabled, Connect collects metrics on a continuous basis for the selected categories as long as Connect Runtime runs. The default setting is enabled. This setting overrides the **Collection Period** setting. **Interval Period**: The period for which Metrics aggregates data per category. For example, if you set **Interval Period** to 1 with the **Interval Period Unit** as hours, and execute channels throughout that hour, there would be a single listing for the "Channel Execution" category with aggregate data for that hour. When an hour has passed from this entry, metrics data is collected and aggregated again towards the next entry.

Interval Period Unit: The unit of time for the interval period. **Collection Period**: Defines the total duration of time for which Connect collects metrics. Connect does not use this setting if **Continuous Collection** is enabled. **Collection Period Unit**: The unit of time for the collection period.

Appendix G: Cron Schedule Job Trigger Configuration

The **Cron Schedule Job Trigger** allows you to create a new **Job Trigger** for any Connector with a **Job** or **DetailedJob** service. From its activation date until its deactivation date, a Cron Schedule Job Trigger executes the desired Job using the configured Cron Expression. For example String Cron Expressions and format information, refer to the Oracle documentation for A Cron Expression.

The **Job Trigger** configuration contains the following properties:

Setting	Description
Trigger Key	Specifies the unique identifier for the current JobTrigger. It cannot match any other JobTriggers in PCR. Format = **Trigger-key-group.Trigger-key-name**, or just **Trigger-key-name** if the JobTrigger belongs in the **DEFAULT** group. **Note** If you just use Trigger-key-name, then any references to this JobTrigger must use **DEFAULT.Trigger-key-name**.
Job Key	Specifies the key to the Job that you want the JobTrigger to execute. Format = **Job-key-group.Job-key-name**, or **DEFAULT.Job-key-name** if the Job key does not have an explicit group. Example Given **Job Key group = "MyJobs"** and **Job Key Name = "Job1"** then the resulting JobKey is **"MyJobs.Job1".**
Description	Specifies the description for this JobTrigger.
Activation Date	Uses an ISO Local format that uses the time zone property in this configuration. If the field is empty, the default is **Now**.
Deactivation Date	Uses an ISO Local format that uses the time zone property in this configuration. If the field is empty, the default is **Never**.
Schedule Cron Expression	Defines the schedule followed by this JobTrigger.
Schedule Time Zone ID	Determines the time zone used by the **Activation Date**, **Deactivation Date**, and **Cron Expression**. If the field is empty, the default is the system's time zone.
Priority	Determines Job execution order when multiple jobs start at the same time. Jobs with higher priorities execute before jobs with lower priorities.

Appendix H: Configure PCR in an active-active configuration

The following section guides you through the steps to setup a cluster that shares the same configuration settings in an SQL server database.

Initial setup

To setup the cluster, complete the following steps.

1. Set up a single PCR node pointing to an SQL server database.
2. Install and configure the connectors.
3. Map a network drive or use a UNC (Universal Naming Convention) path that allows connectors that require access to the local drive to point to a common location that all the load balance instances can access.
4. To setup a load balancer with the single node, complete the following substeps.
    1. Configure the load balancer to use round-robin. We recommend this method as some REST calls just initialize the work and the connections are not kept alive.
    2. Verify the load balancer connects to the service by accessing PCR’s user interface at http://loadbalancer:port.
5. Open the **inserverWorkflow.ini** configuration file in a text editor and update the **connect.url** property to reference the load balancer. For example: **connect.uri =http://loadbalancer:port/rs/workflowTrigger**.
6. Save the file and close the text editor.
7. Verify initial configurations are correct by running a simple test message through each of the connectors.
8. Stop the PCR instance.
9. Copy the installation directory and perform the following substeps on each of your environments.
    1. Paste the copied installation directory into the same location as your source machine.
    2. Execute the following command to register the instance.
       ```
       java -jar connect-runtime-installer-VERSION.jar register
       ```
    3. Start the PCR service.
    4. Verify the PCR service is running by accessing PCR’s user interface at http://localhost:port.
    5. Register the node with the load balancer.
10. Restart the original PCR service once the installation directory has copied over.

Update Content Runtime and Connectors

To update the Content Runtime and Connectors, complete the following steps.

1. Copy the SQL Server database. When the upgrade is complete the original database is no longer in use.
2. Take one PCR service node offline from the load balancer.
3. Stop the PCR service node taken offline.
4. Update the PCR configuration to point to the SQL Server database copy.
5. Start the PCR service.
6. If you are upgrading PCR, complete the following sub-steps:
    1. Stop PCR service again.

   2. Unzip the file and then execute the following upgrade command:

      ```
      java –jar connect-runtime-installer-VERSION.jar upgrade
      ```

      For more information, refer to the Upgrade steps in the Installation guide.
   3. Start PCR service.
7. Update Perceptive Connect Runtime with any connectors you want to update.
8. Stop the PCR service.
9. Copy the installation directory and perform the following sub-steps on each of your environments:
   1. Remove the node from the load balancer.
   2. Stop the service.
   3. Unregister the service.
   4. Backup any H2 databases found in **<installation_dir>/data**. Note that FCC may have its own H2 instance that we recommend you preserve.
   5. Rename the current installation directory to back it up and ensure that you do not overwrite files in the next step.
   6. Paste the copied directory into the same location as it was in the source machine.
   7. Replace the data files.
   8. Register the service with the new PCR jar.
   9. Start the PCR service.
   10. Add the node back to the load balancer.
   11. Verify the load balancer connects to the service by accessing PCR's UI **http://loadbalancer:port**.
10. Start the initial PCR service.
11. Add the initial node to the load balancer.

Known limitations

Configuring the service

Typically, when you configure PCR, the system automatically refreshes the settings. The system knows when modifications are made on the service and refreshes each time. When you are in this shared configuration you must manually trigger the reload. To work around this limitation, we recommend you connect directly to a single service, perform the configuration changes, and then restart the other instances.

Requires low state connector APIs

If the connector adds state that needs to be preserved more configuration to the load balancer may be required to keep the original client connected to the same PCR instance.

The load may not be fairly balanced

The REST calls are just to initialize the work, and the connections are not kept alive. As the work is just distributed to each node one node may be receiving all the largest jobs. Currently, we do not have any health checks to monitor this to keep the load more even.

Stopping services

Some services have long running processes. To take down a server, we recommend you remove the server from the load balancer and let it finish processing its jobs before removing the slave.

Note The system may defer certain jobs, which may rerun on startup of the service. If you want to fully clear the slave, you may need to restart the service to retry failed jobs. For more information, refer to File Conversion Connector.