Perceptive Connect Runtime

Installation and Setup Guide

Version: 1.4.x

Written by: Product Knowledge, R&D
Date: Tuesday, November 22, 2016

Lexmark™ Enterprise Software

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.

- Java SE Runtime Environment, version 1.8 (32-bit or 64-bit)
- A supported web browser
  - Google Chrome
  - Microsoft Internet Explorer, version 11
  - Mozilla Firefox
  - 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 the ZIP file and install Connect Runtime, complete the following steps.

1. Go to the Lexmark Enterprise Software website at www.lexmark.com and log in to the Customer Portal.
2. In the **Product Downloads** page, download the **Perceptive Connect Runtime for Windows** ZIP file.
3. Extract the file to the directory in which you want to install Connect Runtime.

Note If this is an upgrade, extract the file to the Connect Runtime directory you've 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. Open a terminal window and change to the **Connect Runtime directory**.
2. To install Connect Runtime, type `java -jar connect-runtime-installer-VERSION.jar install` and press ENTER.
3. To start the service, type `java -jar connect-runtime-installer-VERSION.jar start` and press ENTER.

Note All data will be stored in **H2 database by default**, if you prefer you may configure Connect Runtime to use Oracle, PostgreSQL or MSSQL. See Configure Database settings to configure the database.

Upgrade the Connect Runtime

You can upgrade any current version of PCR to its next minor version. At this time, it is not recommended that you skip a minor version when upgrading. To upgrade the Connect Runtime, complete the following steps.

1. Open a terminal window and change to the **Connect Runtime directory**.
2. Type `java -jar connect-runtime-installer-VERSION.jar stop` and press ENTER.
3. Type `java -jar connect-runtime-installer-VERSION.jar upgrade` and press ENTER.

- **Note** Before the upgrade, the system creates an automatic backup. Refer to the "Roll back upgrades" topic to restore from the backup.

4. Type `java -jar connect-runtime-installer-VERSION.jar start` and press ENTER.

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 get 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.

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 can be found at http://*{Connect Runtime host name}*:*{port}*. For example, http://localhost:80.

The default user name 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. On the **Upload new bulndles** page, drag and drop the files in the area designated by the box.

- Complete, queued, and processing files display at the bottom of the page. After Connect Runtime processes an item, a summary of the processed file(s) 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:** A list of bundles installed from the file or archive uploaded by the user.
- **Failed:** A list of bundles that failed to install. This includes a message with the reason for the failure.
- **Skipped:** A list of files that PCR can't install, such as non-JAR files.

The affected channel summary

The channel summary contains three items.

- **Fixed:** A count of channels that previously failed to validate that became valid after the installation.
- **Broken:** A count of channels that previously validated that become invalid after the installation.
- **Unknown:** A count of channels that PCR can't 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.

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**. 1. 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. 1. In the **Trigger Information** section, under **Trigger**, select a trigger from the list. 1. 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.

Notes

- 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 would 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**.

- Any validation errors display in a pane to the right of the mapping area.

5. Click **Enable channel**.

View 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.

- On the **PCR Dashboard**, under **Manage**, click **List Channels** or **List Deleted Channels**.
- 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** Connect Runtime's ID for the channel.
- **Name** The user-defined name for the channel. (Legacy channels will display "Unnamed Channel")
- **Trigger Name** The name of the trigger the channel is configured to use
- **Action** The name of the action associated with the channel.
- **Trigger Parameters:** The configuration values entered when initially configuring the channel.

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

Additional columns for List Channels

- **Status** 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 by PCR.
  - A channel is **unsatisfied** if the associated trigger or action are not registered with PCR. You can't enable a channel if its state is unsatisfied.
- **Toggle Status** Additionally, a satisfied channel can be enabled or disabled using the toggle button in the rightmost column.

Additional columns for List Deleted Channels

- **Date Deleted** This is 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:** 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 a channel is expanded on the **List Channels** view, the following information and actions are available.

- **Input and Output Mapping** These boxes 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**
- **Clone**
- **Enable** or **Disable**
- **Delete**

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

- **Input and Output Mapping** These boxes 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

- **Toggle Status** A satisfied channel can be enabled or disabled using this switch.
- **Edit Mapping** If the selected channel is not enabled, this button takes you to the channel's modify mapping page.
- **Clone** If the channel is satisfied, this button takes you to the create channel page and pre-fills some of the source channel's information. Upon saving the new channel, the mapping page will contain a copy of the original channel's input and output mappings.
- **Delete** Clicking this button will delete the selected channel. You will be prompted for confirmation.
- **Restore Channel** Clicking this button will restore the selected channel. You will not be able to 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.

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 `java -jar connect-runtime-installer-VERSION.jar backup` and press ENTER.

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 new connector.

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 `java -jar connect-runtime-installer-VERSION.jar rollback` and press ENTER.
3. Type the numeral of the backup snapshot that you want to rollback to, shown in the list, and then press ENTER.

Appendix B: Customizing and Configuring 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 perceptiveconnectruntime.service`

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 PerceptiveConnectRuntimew.exe.

Connect Runtime properties

The PCR service monitor program provides the following information.

- **General:** Indicates 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:** Indicates the account settings.
- **Logging:** Indicates 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:** Indicates 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.
- **Startup:** Optional. Indicates the Connect Runtime working path. Other settings in this box should remain blank.
- **Shutdown:** Optional. Settings in this box should remain blank.

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 will be connecting to the choices are **"h2", "mssql", "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

an 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, PostgreSQL or Oracle Databases a database must be created to store the PCR data, this can be named 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 Configuration for Linux

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

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

To customize java location

1. In the **linuxEnvironmentFile** file, 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'`**
2. 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.

Configuring 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(s), 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: Set this property to `true` to enable CORS in PCR.
- connect.rs.cors.allow.credentials: Set this property to `true` if your web application needs to pass secure data (such as cookies or authentication headers) to PCR.
- connect.rs.cors.allow.all.origins: Set this property to `true` if you want to allow CORS requests from any external application.
- connect.rs.cors.allowed.origins: This is 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** The port used to accept SSL requests.
- **org.apache.felix.https.keystore** The path for the keystore created in [Create a keystore for PCR].
- **org.apache.felix.https.keystore.password** The password for the keystore.
- **org.apache.felix.https.keystore.key.password** 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** The path for the truststore created in [Create a truststore for PCR].

- **org.apache.felix.https.truststore.password** The password for the truststore. This must be the same as **org.apache.felix.https.keystore.password**.

  Note For further SSL configuration information, see 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 `java -jar connect-runtime-installer-VERSION.jar uninstall` and press ENTER.

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 `java -jar connect-runtime-installer-VERSION.jar register` and press ENTER.

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 `java -jar connect-runtime-installer-VERSION.jar unregister` and press ENTER.

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 ListItem 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** is the string or regular expression it will search for.
- **replace** is 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 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 in to 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 were 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.

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>IntelligentCaptureDocument/InvHeader/INVOICE_
NUMBER</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** is a 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: Configuring 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, see `Configure outbound SSL connections`.

Configure inbound SSL connections

When a machine initiations 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.

- **org.apache.felix.https.enable** Enables SSL in the runtime. Options are TRUE and FALSE.
- **org.osgi.service.http.port.secure** The port used to accept SSL requests. Use any valid port number.
- **org.apache.felix.https.keystore** 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** The password for the keystore.
- **org.apache.felix.https.keystore.key.password** 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** 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** The password for the truststore. This must be the same as the org.apache.felix.https.keystore.password setting.
- **org.apache.felix.https.clientcertificate** Use client authentication on inbound connections. Legal values are `needs`, `wants` and `none`. Default 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 located 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 compelte 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/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.truststore` setting 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 Metrics.

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 of time 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 an 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.