

Pega Robotic Automation Pega Synchronization Server

USER GUIDE

8.0 SP1 and 19.1



© 2021 Pegasystems Inc., Cambridge, MA
All rights reserved.

Trademarks

For Pegasystems Inc. trademarks and registered trademarks, all rights reserved. All other trademarks or service marks are property of their respective holders.

For information about the third-party software that is delivered with the product, refer to the third-party license file on your installation media that is specific to your release.

Notices

This publication describes and/or represents products and services of Pegasystems Inc. It may contain trade secrets and proprietary information that are protected by various federal, state, and international laws, and distributed under licenses restricting their use, copying, modification, distribution, or transmittal in any form without prior written authorization of Pegasystems Inc.

This publication is current as of the date of publication only. Changes to the publication may be made from time to time at the discretion of Pegasystems Inc. This publication remains the property of Pegasystems Inc. and must be returned to it upon request. This publication does not imply any commitment to offer or deliver the products or services described herein.

This publication may include references to Pegasystems Inc. product features that have not been licensed by you or your company. If you have questions about whether a particular capability is included in your installation, please consult your Pegasystems Inc. services consultant.

Although Pegasystems Inc. strives for accuracy in its publications, any publication may contain inaccuracies or typographical errors, as well as technical inaccuracies. Pegasystems Inc. shall not be liable for technical or editorial errors or omissions contained herein. Pegasystems Inc. may make improvements and/or changes to the publication at any time without notice.

Any references in this publication to non-Pegasystems websites are provided for convenience only and do not serve as an endorsement of these websites. The materials at these websites are not part of the material for Pegasystems products, and use of those websites is at your own risk.

Information concerning non-Pegasystems products was obtained from the suppliers of those products, their publications, or other publicly available sources. Address questions about non-Pegasystems products to the suppliers of those products.

This publication may contain examples used in daily business operations that include the names of people, companies, products, and other third-party publications. Such examples are fictitious and any similarity to the names or other data used by an actual business enterprise or individual is coincidental.

This document is the property of:

Pegasystems Inc.
One Rogers Street
Cambridge, MA 02142-1209
USA
Phone: (617) 374-9600
Fax: (617) 374-9620
www.pegasystems.com

Updated: March 5, 2021

Feedback

If you have suggestions or comments for how we can improve our materials, send an email to Robotics-Documentation@pegasystems.com.

CONTENTS

1 The Pega Synchronization Server

- 2 How it works
 - 2 Client-side process
 - 2 Pega Updater Service
 - 3 Pega Loader
 - 4 Server-side process
 - 4 Controlled version distribution
 - 6 Security
- 8 Requirements
- 9 Installing the Pega Synchronization Server
 - 11 Installing the certificate
 - 12 Configuring the server to run within IIS
 - 13 Configuring the server to run under Kestrel
- 14 Configuring the Pega Synchronization Server
 - 14 AppSettingsRepo.json file
 - 16 AppSettingsBundle.json file
 - 16 RepositoryConfig.xml file
- 17 Bundles
 - 17 Importing a bundle
- 18 Configuring a product
 - 18 Activating configuration changes
 - 18 Determining which version to use
 - 18 Fetching a new version
 - 19 Defining the allowed versions
- 19 The product.manifest file
- 21 Viewing the Pega Server Status page

23 Back up configuration files

- 23 Backing up your Pega Synchronization Server configuration

- 23 Backing up product.manifest files

The Pega Synchronization Server

Use the Pega Synchronization Server to store releases for the following Pega products:

- Pega Native Foundation
- Pega RPA Service
- Pega Synchronization Engine
- Pega Robot Runtime

You can download these product releases from [Digital Software Delivery](#) to the Pega Synchronization Server. Then you can load these releases into the Pega Synchronization Server product repository.

This guide includes the following topics:

- [How it works](#)
- [Requirements](#)
- [Installing the Pega Synchronization Server](#)
 - [Installing the certificate](#)
 - [Configuring the server to run within IIS](#)
 - [Configuring the server to run under Kestrel](#)
- [Configuring the Pega Synchronization Server](#)
- [Bundles](#)
 - [Importing a bundle](#)
 - [The product.manifest file](#)
- [Configuring a product](#)
 - [Activating configuration changes](#)
 - [Determining which version to use](#)
 - [Fetching a new version](#)
 - [Defining the allowed versions](#)
- [Viewing the Pega Server Status page](#)
- [Back up configuration files](#)
 - [Backing up your Pega Synchronization Server configuration](#)
 - [Backing up product.manifest files](#)

How it works

The Pega Synchronization Server works with the Pega Synchronization Engine, which is the client-side application, to update your Pega Robot Runtime installations. Security is built into the process to ensure that only Pega-certified content is delivered using this system. For more information, see [Security](#).

With the Pega Synchronization Server, you are in complete control of the software that is downloaded to your desktops. You choose what you want to download and when it is available for download.

The Pega Synchronization Server is the product repository for all versions of Pega Robotic Automation software. With the 19.1 release, Pega Robotic Automation is split into the following products that you update independently:

- Pega Robot Runtime
- Pega Native Foundation
- Pega Synchronization Engine
- Pega RPA Service

When a new version of a product is released, download and install the product for testing on a single, isolated machine. If you choose to take advantage of the new features or the fixes in the new version, you then download and import the new version into the product repository. Client machines then download this update as specified in server-side settings for each product.

Client-side process

Initial installs of the products on client machines are done in the traditional way. Installers are provided for each product and you can install them manually or as part of an automated install process that you build using the provided installers.

The first install on any client machine is called the base install. The base install becomes the lowest possible version of each product that can be installed on that system. The base install includes base versions of the Pega Synchronization Engine, the Pega Native Foundation, the Pega Robot Runtime and, optionally, the Pega RPA Service. After you install the base versions, all future software updates are performed using the Pega Synchronization Engine.

Pega Updater Service

The Pega Updater Service is part of the Pega Synchronization Engine. Upon install it is set to start automatically. When this service is started or restarted it performs the following actions:

- The Pega Updater Service starts using the Local System account.
- The Pega Updater Service Downloader module loads and checks to see if a new version of the downloader module is available on the Pega Synchronization Server.
- If a new downloader module is configured to run, that module is downloaded, the existing downloader module is unloaded and then the new downloader module is loaded by the Pega Updater Service.

- The downloader module downloads the `product.manifest` file for each product from the Pega Synchronization Server. The downloader module checks the `product.manifest` file for each product to determine if any downloads are required. The downloader module then performs any required downloads.
- The Pega Updater Service performs any `PostUpdateActions` that are listed in the `product.manifest` file if the product has updates applied.
- The Pega Updater Service completes each product update by performing the `PostInitializeActions` that are listed in the `product.manifest` file. For the Pega RPA Service and the Pega Native Foundation, this results in their service being started.
- Once the startup process completes, the Pega Updater Service checks its local configuration file for the **PreFetchMins** property value to determine the number of minutes to wait before performing another fetch from the server.
- If the local **PreFetchMins** property value is non-zero, then its value is used. Otherwise, the Pega Updater Service retrieves the value from the Pega Synchronization Server. The Pega Updater Service then waits for the number of minutes specified in the **PreFetchMins** property. When this time expires, the Pega Updater Service polls the Pega Synchronization Server for updates to be downloaded for all the installed products.

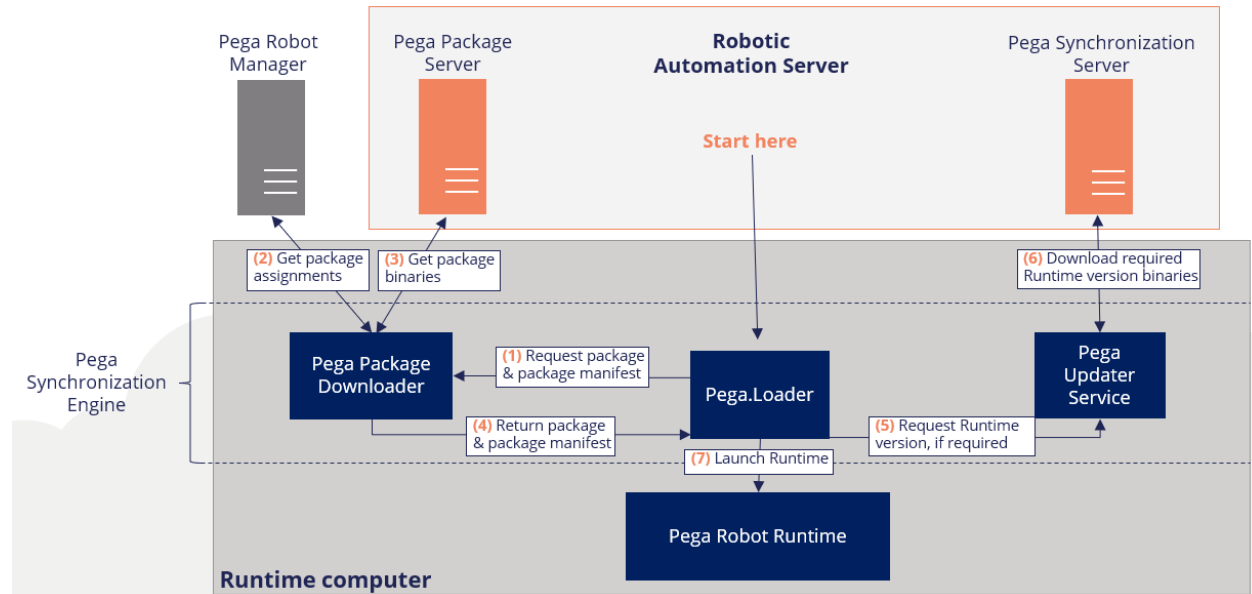
Note	Updates to the Pega Updater Service are only applied when the service is started. Updates to the Pega RPA Service and the Pega Native Foundation service are only applied if those services are not running.
-------------	--

Pega Loader

The Pega Loader application is part of the Pega Synchronization Engine. It coordinates the launch of Pega Robot Runtime. To start Pega Robot Runtime, start the Pega Loader. The Pega Loader performs the following actions:

1. The Pega Loader invokes the Pega Package Downloader to retrieve a package assignment from Pega Robot Manager.
 - If multiple package assignments exist, the Pega Package Downloader prompts for a package choice and then passes the deployment package selected to the next step.
 - If a single package assignment exists, the deployment package is passed to the next step.
2. The Pega Package Downloader determines if the required package is already on the machine and if not, the Pega Package Downloader downloads the package.
3. The Pega Loader reads the `deployment.manifest` file, which is part of the deployment package, to determine the Runtime version that is required to run the package.
4. The Pega Loader invokes the Pega Updater Service to download the required Runtime version if it is not installed on the machine.
 - The Pega Updater Service checks the Pega Robot Runtime cache folder for a sub-folder with the version required.
 - If the required version folder exists, the Pega Updater Service checks for the existence of the `version.ready` file in the version folder. If this file is present, the Pega Updater Service completes processing and returns control to the Pega Loader.
 - If either of the two previous checks fails, the Pega Updater Service requests a changeset and a `version.manifest` file for the required version from the Pega Synchronization Server.

- The Pega Updater Service downloads the files in the changeset if they are not already present in the Runtime cache folder.
- The Pega Updater Service checks each file against the file hash in the `version.manifest` file. If a file hash does not match what is expected, the operation fails.
- The Pega Updater Service writes the `version.ready` file for the version downloaded if all files are downloaded successfully and then returns control to the Pega Loader.
- The Pega Loader starts the executable that is listed in the **LaunchExe** property in the Pega Robot Runtime `product.manifest` file with the path to the deployment package as a command line parameter.



Runtime start process flow

Server-side process

Controlled version distribution

You can distribute new versions when they are added to the server or at a time of your choosing. On the Pega Synchronization Server, there is a `product.manifest` file for each product. This file contains settings that determine when a version is available for download by client machines. These settings can differ for each product. The following table contains common distribution scenarios:

To	See
Test a version on some machines before making it available to the entire population.	Scenario 1
Distribute a new version to all machines prior to activating it.	Scenario 2
Immediately push a new version to all desktops.	Scenario 3

Scenario 1

For this scenario, suppose that your current release of the Pega RPA Service needs to be updated from version 3.1.1.0 to 3.1.2.0, but you want to first test version 3.1.2.0 on several machines.

1. Install the new software directly on the test desktops and perform any required testing.
2. When testing results have been approved, open the `product.manifest` file for the Pega RPA Service product and set the **CurrentVersion** property to "3.1.1.0".
3. Import the Pega RPA Service version 3.1.2.0 bundle into the product repository.
4. Restart the Pega Synchronization Server by performing an admin/reset.

Scenario 2

For this scenario, suppose that your current release of Pega Robot Runtime needs to be updated from 19.1.15.0 to 19.1.24.0 one week from now.

1. Open and edit the `product.manifest` file for Pega Robot Runtime and set the **CurrentVersion** property to "19.1.15.0". Set the **FetchVersions** property to "19.1.24.0".
2. Import the Pega Robot Runtime version 19.1.24.0 bundle into the product repository.
3. Perform an admin/reset on the server to refresh the internal cache.
4. When you are ready to activate the Pega Robot Runtime version 19.1.24.0 for use in production, open the `product.manifest` file for Pega Robot Runtime, and then set the **CurrentVersion** property to "19.1.24.0". Clear the entry from the **FetchVersions** property.
5. Perform an admin/reset on the server to refresh the internal cache.
6. Client machines download version 19.1.24.0 in regularly-scheduled fetches.

When you are pushing product updates for products other than Pega Robot Runtime, restart the client machines so that those machines can receive the product update. Since this is a Pega Robot Runtime update, the update is automatically applied when Pega Robot Runtime is launched.

Scenario 3

For this scenario, suppose that your current release of Pega Native Foundation needs to be updated from 18.09.12101.0 to 18.09.12901.0.

1. Open the `product.manifest` file for Pega Native Foundation and set the **CurrentVersion** property to "latest".
2. Import the Pega Native Foundation version 18.09.12901.0 bundle into the product repository.
3. Perform an admin/reset on the server to refresh the internal cache.
4. Restart the client machines so they can receive the product update.

Security

To prevent a malicious attempt to use the Pega Synchronization Server to place unsafe files on your desktops, the product includes the following security checkpoints:

- **Bundle creation** (Pega Robotic Automation build process) — Before release, when bundles are created:
 - The `version.manifest` file is given a SHA512 digital signature generated by the same private key that is used to sign assemblies. This digital signature validates that the content of this file has not been modified and that the `version.manifest` file was generated during the build process.
 - The `version.manifest` file includes a complete file directory for the bundle with a version hash and a file hash for each file included. Only files that are listed in the `version.manifest` file are processed.
 - The `version.manifest` file is included in the bundle.
- **Bundle import process** (Pega Synchronization Server) — When bundles are imported into the product repository:
 - The signature for the `version.manifest` file is checked using the public key from the primary version file from the product. If the signature does not match, the bundle is rejected.
 - For each file in the bundle the SHA256 hash is checked against the file hash contained in the `version.manifest` file. If the file hash does not match, the bundle is rejected.
 - If any file listed in the `version.manifest` file is missing from the bundle, the bundle is rejected.
 - If files are found in the bundle that are not listed in the `version.manifest` file, those files are not copied to the product repository.
- **Server processes client request for product version information** (Pega Synchronization Server) — When the server receives a request for product version information, it prepares a changeset. A changeset is a list of files that have changed between the version requested and a previous version. The system uses the changeset to determine which files must be downloaded from the server to run the requested version. The only files that the system downloads are the files that differ from the files that are already present on the machine.
 - The signature for the `version.manifest` file is checked using the public key from the primary version file from the product. If the signature is invalid, the request fails.
 - The changeset sent to client contains the file hash from the `version.manifest` file for each file in the changeset which the client uses to validate each file.
- **Client requests product version information** (Pega Synchronization Engine/Pega Updater Service) — When the client receives a version changeset request:
 - A changeset is retrieved which contains a file hash for each file. The system retains this information in memory for the life of the current update process.
- **Server processes client request for file download** (Pega Synchronization Server) — When the server receives a file download request:
 - The signature for the `version.manifest` file is checked using the public key from the primary version file from the product. If the signature is invalid, the request fails.
 - The file requested is checked to ensure that it has not been tampered with by retrieving the file hash from the `version.manifest` file and comparing it with the file's SHA256 hash before transmitting the file to client. If the SHA256 hash is invalid, the request fails.

- **Client requests file download** (Pega Synchronization Engine/Pega Updater Service) — When the client receives a file download request:
 - Each file download that is requested by the Pega Updater Service holds the file requested in memory and compares the file's SHA256 hash against the file hash from the changeset. If the hash check fails, the update process fails and the file is not written to the disk.
 - The product cache folder is in the install folder for the Pega Synchronization Engine. The default location is in the %ProgramFiles% or %ProgramFiles (x86)% folders and which should be secured by the appropriate Windows permissions.
- **Client launches product** — (Pega Loader)
 - The Pega Loader application validates the build signing certificate of the executable that it is to launch and only launches valid Pega product executables.

Requirements

To use the Pega Synchronization Server, the machine should meet the following requirements:

Component	Requirement
Processor *	Dual core or faster processor with a minimum speed of 2 gigahertz (GHz)
RAM	8 gigabytes (GB) total RAM
Disk space	1 terabyte (TB)
Operating system	These 64-bit operating systems are supported: Windows 7 SP1 ** Windows 8.1 Windows 10 Windows Server 2008 R2 ** Windows Server 2012 R2 Windows Server 2016
Other software	.NET Core 2.2.3 Runtime and Hosting Bundle for Windows Internet Information Services v10

* If you are also using the Robotic Automation Security Token Service (STS), you need a quadruple core or faster processor with a minimum speed of 2 GHz.

** Ensure that you have applied the Microsoft [KB 2533623](#) hotfix.

Note	Install the Pega Synchronization Server, the Pega Package Server, the Pega Server Status page and the Robotic Automation Security Token Service (STS) on the same server. For more information, see the Pega Robotic Automation Security Token Service User Guide .
-------------	---

Installing the Pega Synchronization Server

Use the Pega Synchronization Server to store Pega product releases. The Pega Synchronization Server Setup wizard is included in the Pega Robot Runtime download. You can install this application on any machine that resides inside your organization's network that meets the [requirements](#).

Before you begin

- If it is not already installed, install the Microsoft .NET Core 2.2.3 Runtime and Hosting Bundle for Windows. This software is included in the download.
- If it is not already present, add the SSL certificate to the Local Machine Personal or Web Hosting certificate store.

You can configure the Pega Synchronization Server to run using Internet Information Services (IIS) or the Kestrel web server. Using IIS has the following advantages:

- IIS runs as a service, so it starts running when Windows starts.
- IIS allows for response compression which increases the speed of downloads and reduces network traffic.

Installation

To install the Pega Synchronization Server, perform the following steps:

1. Use [Digital Software Delivery](#) to download the Pega Robot Runtime setup file. Unzip this file into a temporary folder location, locate the `ServerInstalls.zip` file.

Note For best results, install the software from a local drive, not a mapped drive.

2. Unzip the `ServerInstalls.zip` file and locate the `PegaSynchronizationServerSetup.exe` file.
3. Right-click the setup executable (`PegaSynchronizationServerSetup.exe`) and select **Run as Administrator**. The Setup wizard starts.
 - For IIS installations, the recommended installation path is: `C:\Intepub\Pega Synchronization Server`.
 - For Kestrel, the recommended installation path is: `C:\Program Files (x86)\Pegasystems\Pega Synchronization Server`.
4. Update the contents of the `appsettingsRepo.json` file in the installation folder.
 - In the **Kestrel** section, enter the **EndPoints > Https > Url** property, for example:

```
"Kestrel": {
  "EndPoints": {
    "Http": {
      "Url": "http://*:5001"
    },
    "Https": {
      "Url": "https://*:5004"
    }
  }
},
```

- In the **Certificates** section, enter the **Subject**, **Store**, and **Location** properties, for example:

```

"Certificates": {
  "Default": {
    "Subject": "SSLCertName",
    "Store": "My",
    "Location": "LocalMachine",
    "AllowInvalid": false
  }
}

```

- In the **Products** section, enter the location of the product repository in the **Directory** property, for example:

```

"Products": {
  "Directory": "C:\\Updater\\Products",
  "DebugSymbols": false,
  "MemoryCacheDurationHours": 48
},

```

5. In the `web.config` file in the installation folder, add the full path to the `dotnet.exe` file that was installed with the Microsoft .NET Core 2.2.3 Runtime and Hosting Bundle for Windows in the **processPath** property. By default, this file is installed in the `C:\Program Files\dotnet` folder. The following is an example:

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <!-- For IIS hosting appsettingsRepo.json configuration file needs to have:
    * Kestrel|Certificates section configured to point at a matching to IIS application certificate
    * HttpServer|Https port configuration value adjusted to match IIS application endpoint port
  -->
  <!-- Configure your application settings in appsettings.json. Learn more at http://go.microsoft.com/fwlink/?LinkId=786380 -->
  <system.webServer>
    <handlers>
      <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModuleV2" resourceType="Unspecified" />
    </handlers>
    <aspNetCore processPath="C:\Program Files\dotnet\dotnet" arguments=".\Pega.ProductRepository.dll" stdoutLogEnabled="false" stdout>
  </system.webServer>
</configuration>

```

6. Create the product repository folder that you specified in the **Directory** property in the `appsettingsRepo.json` file, for example: `C:\Updater\Products`.
7. Follow the installation guidelines for Kestrel or IIS, depending on the service you use.

Installing the certificate

Complete this step before you configure IIS or Kestrel. An SSL certificate is required to secure all transmissions between client and server.

You can use an existing SSL certificate if your organization has one in place. If you need a new certificate, the SSL certificate must be issued by a certificate authority, so you have a quick way through certificate revocation to signal if your private key becomes compromised.

Each certificate authority has a different process for generating a certificate with a private key. Confirm with your certificate authority the exact process that is required. Typically, a certificate request is issued from IIS and uploaded to the certificate authority through their website. Their response is then imported back into IIS from the same machine that issued the request. This completes the transaction.

Some certificate authorities have their own software package that you install and use to request and generate a certificate. For security, make the certificate requests on the machine that uses the certificate, so the private key does not need to be exported or transmitted.

To use a certificate that is generated on a different machine than the one that hosts the Pega Synchronization Server, perform the following tasks:

- Export the certificate with its private key.
- Choose a temporary password to encrypt the private key contents.
- Import the key onto the machine that is hosting the Pega Synchronization Server in the Local Machine Trusted Root Certification Authorities certificate store by using the Microsoft Management Console (`MMC.exe`).

Perform the following steps:

1. On the machine that has the certificate, start the `MMC.exe` program and add the Certificates Snap-In for the certificate store that contains the certificate.
2. Select the certificate and choose the option to export it.
3. Follow the instructions in the Certificate Export wizard, selecting to export the private key. check the following options:
 - Include all certificates in the certification path if possible
 - Export all extended properties.
4. Set a temporary password for encrypting the private key.
5. Save the Personal Information Exchange (`PFX`) file and transfer that file to the machine that hosts the Pega Synchronization Server.
6. On the machine that hosts Pega Robot Runtime, start the `MMC.exe` program and add the Certificates Snap-In for the computer account of the local computer.
7. In the Trusted Root Certification Authorities certificate store, import the saved certificate and enter the temporary password. Follow these guidelines:
 - Do not mark the private key as exportable. This prevents the key from being copied from your computer.
 - Select the **Include all extended properties** option.

Configuring the server to run within IIS

To configure the Pega Synchronization Server to run within IIS, complete the following steps:

1. Open the IIS Manager.
2. On the **Connections** tab, right click the `Sites` folder, and then select **Add Website**.
3. In the Add Website dialog box, configure the website parameters:
 - In the **Site Name** field, enter **Pega Synchronization Server**.
 - In the **Physical Path** field, specify the install folder.
 - In the **Binding Type** field, enter **https**.
 - In the **Binding Port** field, enter the port that you want to use. Ensure that your entry does not conflict with other applications.
 - In the **SSL Certificate** field, from the list of already installed certificates, select the CA certificate that you want to use.
4. Click **OK**, and then expand **Sites > Pega Synchronization Server**.
5. Double-click **Request Filtering** and ensure that the following file name extensions are allowed:

.class	.config	.crx	.dll	.exe
.ini	.json	.pak	.pdb	.sys
.tlb	.txt	.xml	.xpi	

6. On the **Connections** tab, click **Application Pools**.
7. In the **Application Pools** list, right-click **Pega Synchronization Server**, and then select **Basic Settings**.
8. In the Edit Application Pool dialog box, set .NET CLR Version to **No Managed Code**.
9. Click **OK**.
10. On the **Connections** tab, right-click **Pega Synchronization Server**, select **Manager Website**, and then click **Start**.
11. In the installation folder, edit the `appsettingsRepo.json` file, by completing the **Certificate** section with information that describes the certificate that you configured in the **SSL Certificate** field.

Configuring the server to run under Kestrel

To configure the Pega Synchronization Server to run on a Kestrel server, complete the following steps:

1. Edit the `AppSettingsRepo.json` file in the install folder. For more information, see [AppSettingsRepo.json file](#).
 - Enter the Https URL.
 - Select the certificate used to secure the connection.
2. Create and edit a command file. The following is an example:

```
cd {install folder}
start dotnet Pega.ProductRepository.dll --urls https://*:{port}
```

3. Place a shortcut to the command file in the default **Startup** folder so that it starts when anyone logs into the machine. For example:
C:\Users\Default\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup
4. Run the command file.

When you are running the Kestrel web server, ensure that the system is logged in. Logging out of the system stops the web server. The web server does not restart automatically when you reboot the server. Ensure that the system is logged in before the web server starts.

Configuring the Pega Synchronization Server

The Pega Synchronization Server has the following files in which you perform configuration tasks:

- [AppSettingsRepo.json file](#) – The settings file for the Pega Synchronization Server.
- [AppSettingsBundle.json file](#) – The settings file for the BundleImport program.
- [RepositoryConfig.xml file](#) – The logging configuration file.

These files are located in the installation folder.

AppSettingsRepo.json file

Use this file to configure the Pega Synchronization Server. This file contains the following sections:

- [AllowedHosts](#)
- [Logging](#)
- [Kestrel](#)
- [HttpServer](#)
- [Server](#)
- [Products](#)
- [Fetch](#)

You do not have to modify or make entries in all sections.

AllowedHosts

This setting enables Host Filtering Middleware when you use a Kestrel web server. This setting is not applicable and has no effect if you are using IIS.

Logging

Use this setting to control the amount of logging information that is written to external logs. You have the following choices for the log levels, in descending verbosity: Trace, Debug, Information, Warning, Error, Critical, and None.

Kestrel

The Kestrel section includes the following sections:

Section	Description
Https	When you use a Kestrel web server, use this section to configure the server endpoint. Enter the Https URL in this format: <code>https://*: {port}</code> For example: <code>https://*:5004</code> . This section is ignored if you use IIS.

Section	Description
Certificates	<p>Specify the CA certificate that is used to secure communication. This section is required for both Kestrel and IIS servers.</p> <ul style="list-style-type: none"> Subject — Enter the subject name for the certificate. Store — Specify the certificate store from which to load the certificate. For example: Root or My. Location — Specify the location of the store from which you want to load the certificate. For example: LocalMachine or CurrentUser. AllowInvalid — Enter True to indicate if the system should consider invalid certificates, such as self-signed certificates.

HttpServer

Not used.

Server

The Server section includes the following properties:

Property	Description
MaxConnections	Specify the number of connections allowed for fetch processing. On-demand requests are always accepted. If you enter zero (0), there are no restrictions on the number of sessions.
ResponseCache DurationLongMins	The internal system timeout. Do not change this property unless directed by Pega Support.
ResponseCache DurationShortSecs	The internal system timeout. Do not change this property unless directed by Pega Support.
ResponseCompression	Enter False . For Kestrel servers, this property should always be set to False. For IIS servers, enable response compression in IIS.

Products

The Products section includes the following properties:

Property	Description
Directory	Enter the path to the product repository. Indicate backslashes with two backslashes. For example: C:\Updater\Products.
DebugSymbols	<p>Enter False. This property determines whether <code>pdb</code> files are sent to clients for debugging purposes. These files are large and are only needed when debugging.</p> <p>In most cases, instead use the client setting that allows a single client to download the Debug Symbols.</p>
MemoryCache DurationHours	The internal system timeout. Do not change this property unless directed by Pega Support.

Fetch

The Fetch section includes the following properties:

Property	Description
PreFetchMins	Enter the default frequency, in minutes, that the pre-fetch cycle should occur on the client machines. You can override this setting for individual clients. It is recommended that you set this property to allow the client to connect with the server two or three times a day.
MaxPreFetchSessions	Enter the maximum number of concurrent pre-fetch sessions allowed by the server. Use this setting to prevent congestion on the server when you are distributing a large update.
PreFetchSessionTime outMins	Enter the maximum time, in minutes, before a pre-fetch session times out and the session token is released.

AppSettingsBundle.json file

Use this file to control the amount of logging information that the BundleImport program produces. The system writes the logging information to the `BundleImport.log` file in the `%programdata%\Pegasystems\Updater` folder. The `AppSettingsBundle.json` file is located in the Pega Synchronization Server install folder.

Section	Description
LogLevel	Use this section to control the log levels for the log files and console. You have the following choices for the log levels, in descending verbosity: Trace, Debug, Information, Warning, Error, Critical, and None.
Console	Use this section to control the amount of logging information that is written to the console window. You have the following choices for the log levels, in descending verbosity: Trace, Debug, Information, Warning, Error, Critical, and None.

RepositoryConfig.xml file

Use this file to determine how much space is allocated for the logging of the BundleImport and the product repository applications. For example, use the properties in this file to specify the maximum file size and the number of roll backups to retain. This file is in the Pega Synchronization Server install folder.

These properties apply to both the `BundleImport.log` and the `ProductRepository.log` files.

Property	Description
maximumFileSize	Specify the maximum file size for a log file. Specify the maximum size with the suffixes "KB", "MB," or "GB" so that the size is interpreted as kilobytes, megabytes, or gigabytes.
maxSizeRollBackups	Specify the total number of log files that are permitted. To change the number of roll backups to keep, update this property.

Bundles

Bundles are available from Digital Delivery for every released version of each product. The Pega Robot Studio installation download includes a `Bundles.zip` file. It is not necessary to import every version into the Pega Synchronization Server. The following table provides recommendations:

Note	<p>To get the most current bundles, download the latest build of Pega Robotic Automation. View the build notes to see the version numbers of the individual bundles that are delivered with each build.</p> <p>You must import the oldest base-client Pega Robot Runtime and Pega Synchronization Engine versions across the environment bundle to the Pega Synchronization Server. For maximum efficiency, for each Pega Robot Runtime base version and Pega Synchronization Engine base version, ensure that the same base versions are also installed on the Pega Synchronization Server.</p>
-------------	--

Product	Recommendation
Pega Robot Runtime	Each version of Pega Robot Studio that was used to publish a package should have a corresponding version of Runtime imported to the Pega Synchronization Server.
Pega Native Foundation	Stay current with the most recent release of the Pega Native Foundation. This product updates automatically when you restart the Pega Updater Service.
Pega RPA Service	Stay current with the most recent release of the Pega RPA Service if you use RPA. This product updates automatically when you restart the Pega Updater Service.
Pega Synchronization Engine	Stay current with the most recent release of the Pega Synchronization Engine. This product updates automatically when you restart the Pega Updater Service.

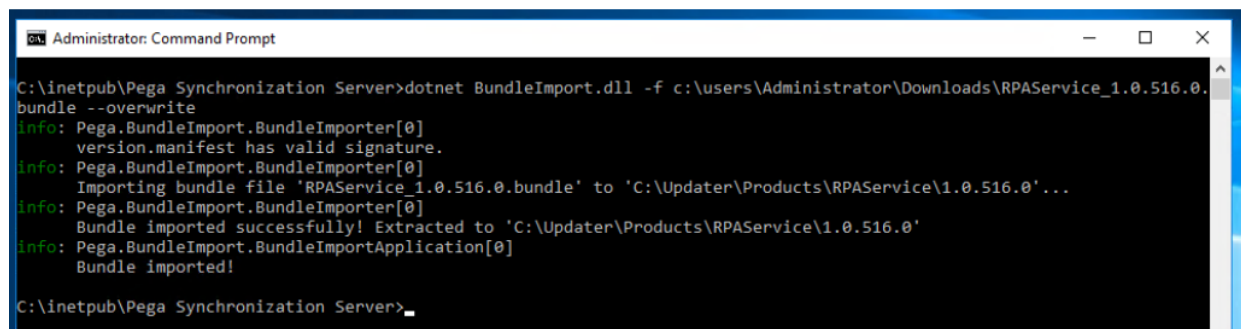
Importing a bundle

Bundles are downloaded from [Digital Software Delivery](#) to the Pega Synchronization Server.

Import a bundle from a command prompt using the `BundleImport.dll` file that is located in the Pega Synchronization Server install folder. Run the command from the install folder, using the following command syntax:

```
dotnet bundleimport.dll --file %pathtobundle% --overwrite (optional)
```

The following is an example:

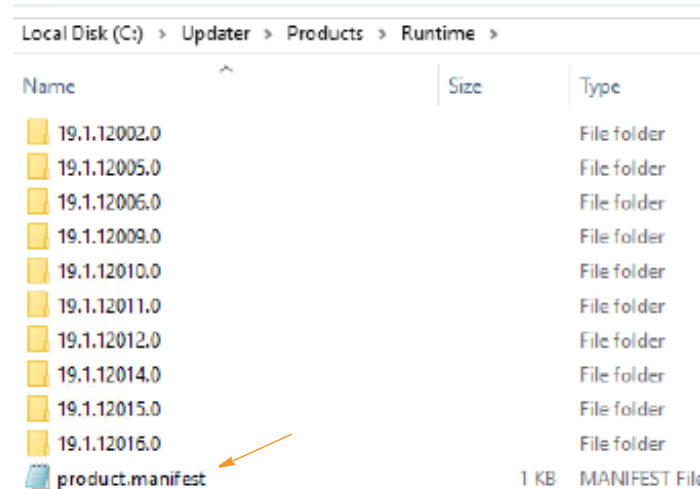


```
Administrator: Command Prompt
C:\inetpub\Pega Synchronization Server>dotnet BundleImport.dll -f c:\users\Administrator\Downloads\RPAService_1.0.516.0.bundle --overwrite
info: Pega.BundleImport.BundleImporter[0]
      version.manifest has valid signature.
info: Pega.BundleImport.BundleImporter[0]
      Importing bundle file 'RPAService_1.0.516.0.bundle' to 'C:\Updater\Products\RPAService\1.0.516.0'...
info: Pega.BundleImport.BundleImporter[0]
      Bundle imported successfully! Extracted to 'C:\Updater\Products\RPAService\1.0.516.0'
info: Pega.BundleImport.BundleImportApplication[0]
      Bundle imported!
C:\inetpub\Pega Synchronization Server>
```

The `BundleImport.log` file is placed in the `%appdata%\Pegasystems` folder by default. This file logs all activities that occur when you run the `BundleImport.dll` file. Review this log file to verify which bundles were loaded and any exception messages generated by an invalid bundle load.

Configuring a product

For each product, you can configure which versions are available for your users to download and which versions the users should run. This information in the `product.manifest` file controls this behavior. There is one `product.manifest` file for each product and the file is located in the root folder of each product in the product repository.



For a description of each property in this file, see [The product.manifest file](#).

Activating configuration changes

The changes made to the `product.manifest` file are not recognized by the Pega Synchronization Server until you perform an admin/reset. Click **Reset** on the Server Status page to perform an admin/reset.

Determining which version to use

To control which version of a product is in use, set the **CurrentVersion** property to one of the following options: oldest, latest, or a specific version number. When the Pega Updater Service restarts on a client machine, this value is read and the specified version of the product is loaded. For the Pega Robot Runtime product, this setting is overridden by the deployment package version.

Fetching a new version

To cause a user to pre-fetch a product version that you plan on using in the future, enter the version number in the **FetchVersions** property. You can enter multiple versions separated by commas, for example: 19.1.1000,19.1.1001,19.1.1002.

Use this feature to allow Pega Robot Runtime systems to stagger their version downloads and avoid a heavy network load when switching versions. You can pre-fetch versions and then activate those versions using the **CurrentVersion** property after most systems download the new version.

Defining the allowed versions

Use the **OldestVersion** and **LatestVersion** properties to restrict access to a limited number of versions. Both settings accept the oldest, latest, or specific version numbers.

- By setting the **OldestVersion** property to a specific number (typically your base install version), you can guarantee that older versions are not used.
- By setting the **LatestVersion** property to the latest version that you have tested, you can prevent a newer version that is loaded into the product repository from being downloaded to client machines.

Note Setting the **OldestVersion** property to a value higher than your base install can prevent some deployment packages from loading.

The product.manifest file

This file contains product-specific settings to control how the Pega Synchronization Server and the Pega Updater Service behave for a specific product. This file is placed in each product's folder by the BundleImport program. You can modify this file to change the behavior of a product.

Property	Description
Name	Specifies the product name.
LaunchExe	Specifies the application that Pega Loader runs when the product is started. For example, this setting is either <code>OpenSpan.Runtime.exe</code> or <code>OpenSpan.AgileDesktop.exe</code> for the Pega Robot Runtime product.
LaunchArgs	Specifies the command line arguments that the Pega Loader uses when the product starts. For more information, see the Pega Robotic Automation Installation Instructions .
AlwaysCopy	Do not change this setting unless directed to by Pega Support. Determines if a product can download a partial set of files for a version.
CurrentVersion	Defines the current version of the product to be run. You can enter "latest", "oldest," or a specific version number, such as "1.0.48.0". Enclose your entry in quotation marks ("entry"). For Pega Robot Runtime, this setting is overridden by the deployment package version, but for all other products this setting determines which version is launched.
LatestVersion	Defines the latest version of the product to be run. Use this setting to restrict access to versions that are not ready to be released. You can enter "latest", "oldest," or a specific version number, such as "1.0.48.0". Enclose your entry in quotation marks ("entry").
OldestVersion	Defines the oldest version of the product to be run. Use this setting to restrict access to versions that are no longer to be used. You can enter "latest", "oldest," or a specific version number, such as "1.0.48.0". Enclose your entry in quotation marks ("entry").

Property	Description
FetchVersions	Defines the versions that you want to download to client machines. The Pega Updater Service on the client machine downloads the versions in the background. Separate the versions with commas and enclose them in quotation marks. Here is an example: "1.0.48.0","1.0.49.0, 1.0.50.0".
PostUpdateActions	A json array of actions to perform after a product is updated. For more information, see Supported actions .
PostInitializeActions	A json array of actions to perform after a product update actions are completed. For more information, see Supported actions .

Supported actions

Note Only modify the following settings under the direction of Pega Support.

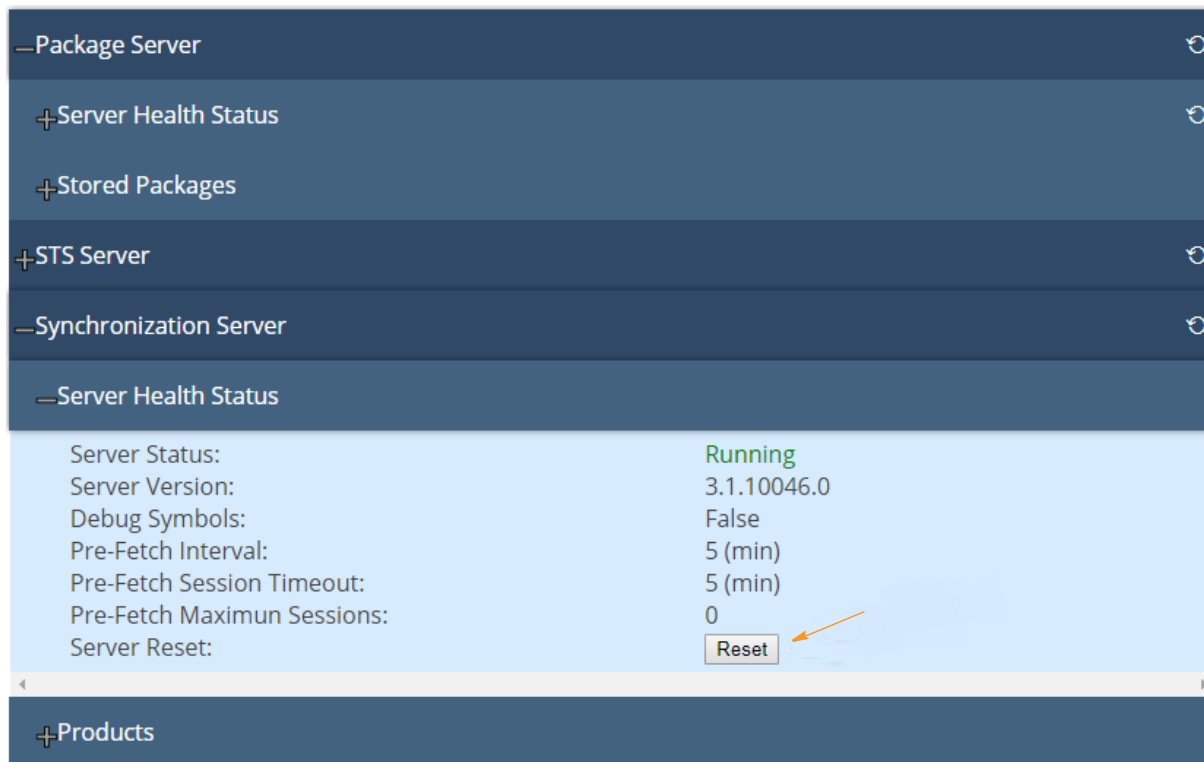
The `product.manifest` file supports several actions that are triggered after an update is applied to a product (PostUpdateActions) or after the Pega Updater Service completes its initialization on startup (PostInitializeActions). You can include the following supported actions in the PostUpdateActions or PostInitializeActions properties:

- **Update_service** — Updates the Windows registration for the Pega Notification Manager Service, the Pega RPA Service, or the Pega Updater Service after the Pega Synchronization Engine has downloaded an update for one of these services.
- **Start_service** — Starts a service (Pega Notification Manager Service or Pega RPA Service) after the Pega Updater Service has completed its initialization at startup.
- **Run_Program** — Runs a program after the Pega Updater Service initialization at startup has completed.

Viewing the Pega Server Status page

Use the Pega Server Status page to monitor the Pega Package Server, the Robotic Automation Security Token Service, and the Pega Synchronization Server. Load the Pega Server Status page by going to the URL and port assigned to the page.

On this page you can see basic server health settings for the Pega Synchronization Server. The following is an example:



Click **Reset** to start an Admin Reset.

Expand the Products section to view the current `product.manifest` file settings as well as the versions that are available in the product repository.

+

NativeFoundation

▢

RPAService

▢

Manifest

```
{
  "name": "RPAService",
  "launchExe": "Pega.RpaService.Service.exe",
  "launchArgs": "",
  "currentVersion": "3.1.10014.0",
  "fetchVersions": [],
  "postUpdateActions": [
    {
      "ServiceName": "PegaRPAService",
      "ServiceExe": "Pega.RpaService.Service.exe",
      "Name": "UpdateRPAService",
      "Type": "update_service",
      "Enabled": true
    }
  ],
  "postInitializeActions": [
    {
      "ServiceName": "PegaRPAService",
      "StartTimeout": 10000,
      "Name": "StartRPAService",
      "Type": "start_service",
      "Enabled": true
    }
  ],
  "splashScreen": false
}
```

▢

Versions

Version: 3.1.10007.0

Version: 3.1.10008.0

Version: 3.1.10009.0

Version: 3.1.10010.0

Version: 3.1.10011.0

Version: 3.1.10012.0

Version: 3.1.10013.0

Version: 3.1.10014.0 (Current)

+

Runtime

Back up configuration files

The Pega Synchronization Server is a repository for the product updates that Pega periodically releases. These updates are available from [Digital Software Delivery](#) and can be downloaded multiple times if needed.

To protect your organization from a loss of data, you can choose to back up these updates.

The following configuration files are used by the Pega Synchronization Server:

- The configuration file for the Pega Synchronization Server — `appsettingsRepo.json`
- The configuration file for the BundleImport program — `appsettingsBundle.json`
- The logging configuration file — `RepositoryConfig.xml`
- The `product.manifest` files for each product

Back up these configuration files whenever you make changes to them. For more information, see the following topics.

Backing up your Pega Synchronization Server configuration





1. In Windows Explorer, go to the Pega Synchronization Server installation folder.
2. Make a copy of the `appsettingsRepo.json` file, the `appsettingsBundle.json` file, and the `RepositoryConfig.xml` file.
3. Store the backup copy of the `appsettingsRepo.json` file, the `appsettingsBundle.json` file and the `RepositoryConfig.xml` file where you store other system backups.

If you need to reinstall the Pega Synchronization Server, replace the installed files with these copies.

Note If the Pega Synchronization Server is running when you replace the files, restart the program. For more information, see [Activating configuration changes](#).

Backing up product.manifest files

1. In Windows Explorer, go to the folder that you defined in the **Products > Directory** property in the `appsettingsRepo.json` configuration file.
2. Make a copy of the `product.manifest` file in each subfolder.
3. Store the backup copies of the `product.manifest` files where you store other system backups. Use the same folder structure to store the files that the product repository uses.

Name	Date modified	Type
 NativeFoundation	4/18/2019 10:42 AM	File folder
 RPAService	4/18/2019 10:36 AM	File folder
 Runtime	4/22/2019 1:13 PM	File folder
 Updater	4/18/2019 10:35 AM	File folder

To reinstall the `product.manifest` files, copy the files from this folder into the corresponding folder location defined in the **Products > Directory** property in the `appsettingsRepo.json` configuration file.

Note If the Pega Synchronization Server is running when you replace the files, restart the program. For more information, see [Activating configuration changes](#).
