Perceptive Enterprise Deployment Suite

Getting Started Guide

Version: 1.3.x

Written by: Product Knowledge, R&D
Date: June 2021

Copyright

Table of Contents

Copyright **2**
What is Perceptive Enterprise Deployment Suite? **5**
What is the PEDS Server? **5**
Configure parent and child PEDS Servers 6
What is the PEDS Client service? **6**
Configure settings for the Client service 6
What is the PEDS Notifier? **7**
What is the PEDS Launcher? **7**
What is the PEDS Management Console? **7**
What is an update package? 7
Create an update package *8*
Import an update package *8*
Modify an update package component *9*
Add a configuration file to an update package *9*
Add a file to an update package *10*
Add an installer file to an update package *10*
Add a patch file to an update package *11*
Add a script to an update package *11*
What is a deployment plan? 11
What are trigger types? 12
What are update types? 12
What are deployment types? 12
Create a deployment plan *12*
About validating an update package 14
About publishing an update package 14
PEDS Launcher configuration settings 14
What are PEDS reports? **14**
Client Product Version Report - ImageNow 15
Client Product Version Report – Interact Desktop 16
PEDS Version Report 16
Appendix A: Using a command line interface with PEDS **18**
Create an update package using commands 18
Validate an update package using commands 18

Create a deployment plan using commands 18
Modify a deployment plan component using commands 19
Management Console commands 19
Update package commands 20
Deployment plan commands 22
PEDS Notifier commands 24
PEDS Launcher 24
Appendix B: XML file settings 25
Update package 25
Syntax 25
Variables 25
Installer attribute values 26
Installer 26
Executable 26
Command line parameters 26
File attribute values 26
useSourcePath element 26
Deployment plan 27
Deployment_plan.xml settings 28
Settings for an installed update 28
Settings for a staged update 29

What is Perceptive Enterprise Deployment Suite?

Perceptive Enterprise Deployment Suite (PEDS) is a tool you can use to distribute updates to users' computers automatically. PEDS allows you to easily create customized update packages and stage their deployments.

To use PEDS, you create an update package and a deployment plan, and then publish both the update package and deployment plan to the PEDS Server. The update package includes all of the contents of the update. The deployment plan designates how and when the update package is pulled from the PEDS Server. Once a package is published, the users’ computers pull the update package files from the PEDS Server and store them locally until the installation parameters set in the deployment plan are met. The update package files will run after the deployment plan parameters are met.

PEDS also includes a reporting feature. These reports give information on the state of the computers at large in respect to Perceptive Software products and versions. PEDS reporting currently provides information for ImageNow, Interact Desktop, and PEDS.

You administer, configure, and establish parameters for PEDS using the following tools:

- PEDS Server
- PEDS Client Service
- PEDS Notifier
- PEDS Launcher
- PEDS Management Console
- PEDS Reports

What is the PEDS Server?

The PEDS Server is a servlet that hosts the PEDS update packages.

The PEDS Server monitors a directory for changes to files and makes updates available for download. You can configure settings, such as logging levels and when to check for new update packages, and set the directories to use for file storage within the pedserver.ini file.

You can configure the PEDS Server as a child of another PEDS Server in the pedserver.ini file. Child servers also monitor their parent and download update packages to their own monitored directory.

The monitored directory is specified by the update.directory setting in the pedserver.ini file. The directory structure includes the following structure:

- <update.directory>\<appID>\deployment_plan.xml
- <update.directory>\<appID>\<version>\config\
- <update.directory>\<appID>\<version>\install\
- <update.directory>\<appID>\<version>\updater\

Configure parent and child PEDS Servers

A child server is an additional server in your system that communicates with the parent server. To configure parent and child servers in PEDS, complete the following steps.

Privileges You must have System Administrator privileges to perform this function.

Note You designate the parent server by entering its URL in the locations specified below. In the pedserver.ini file of the parent server, these fields will be empty. Only child servers will have values for these settings.

1. From the location where your child PEDS Server servlet is stored, click **pedserver**>**WEB-INF**>**etc** and open the pedserver.ini file.
2. Locate the `parent.server.url` setting under the `[General]` heading and enter the URL of the parent server for your system.
3. Locate the `parent.reporting.server.url` setting under the `[Reporting]` heading and enter the URL of the parent reporting server for your system.

What is the PEDS Client service?

The PEDS Client service is a Windows service on a user's computer. This service runs in the background and coordinates the various actions that take place during an update.

The PEDS Client Service also controls the communication that takes place between a user’s computer and the PEDS Server. This service interprets the deployment plan and tells the user’s computer when to pull the update package from the PEDS Server, and when to install the update.

Configure settings for the Client service

To configure settings for the Client service, complete the following steps.

1. Open the **PEDClientService.exe.config** file using the appropriate editor.
2. In the **<applicationSettings>** section, make changes to the configuration settings in your file. Refer to the “Client service settings” topic for descriptions of each setting. The settings in the <applicationSettings> section are:
   - logDirectory
   - logLevel
   - checkIntervalInSeconds
   - ignoreServerConfig
   - stagingLocation
3. Save the file.

What is the PEDS Notifier?

The PEDS Notifier displays messages indicating the process or status of updates. When it is not displaying a message, the PEDS Notifier appears as an icon that resides in the system tray of a user's computer.

The PEDS Service Monitor starts the PEDS Notifier when the PEDS Client Service receives the command to start an installation. If you close the dialog box, the PEDS Notifier is minimized to the system tray.

What is the PEDS Launcher?

The PEDS Launcher is a shortcut to your target application or applications. Currently, it can be used to open ImageNow, Perceptive Content, and Interact Desktop. The PEDS Launcher checks for updates prior to launching the application when a user clicks on the application's icon. If updates are available, they are pulled from the PEDS Server. If updates are not available, the PEDS Launcher opens the target application.

The executable file that contains the launcher parameters and settings is PEDLauncher.exe. The application that the PEDS Launcher starts, or checks for updates for, can be indicated either by command-line arguments or by a configuration file. Command-line parameters take precedence over configuration file settings.

What is the PEDS Management Console?

You can use PEDS Management Console to configure update packages and deployment plans, as well as publish update packages according to the deployment plan parameters.

There are two tabs in the PEDS Management Console: Package Generator and Deployment Plan. Update packages are created on the Package Generator tab and Deployment plans are configured on the Deployment Plan tab. Publishing is also done from the Deployment Plan tab.

What is an update package?

An update package is a collection of files that contains installation files, installer attributes, and any other files or scripts you want to use to update an application. An XML file indicates where all of the files in the update package are located.

The required elements of an update package are the application identifier and the version that you are updating, the path to the directory that contains the update installer, and the installation files. The following are optional update package components:

- Configuration files
- Installer files
- Other files
- Patch files
- Scripts

Create an update package

An update package is an XML file that contains the installation files and the installer attributes you use to update an application. To configure the contents of an update, complete the following steps.

1. Click **Start** > **All Programs** > **Perceptive Software** > **PEDS Management Console**.
2. In the **PEDS Management Console**, on the **Package Generator** tab, in the **Select application** list, choose the application to update.
3. Under **Package Library**, click **New**.
4. In the **New Package** dialog box, the **Version number** field, enter a version number for the version of the application that the update is installing. The version number must match the version of the application that the package installs.
5. Optional. In the **Revision number** list, enter a revision number. You can use this number to differentiate between revisions of packages with the same version number.
6. Under **Package Components**, click **Add**. A list of components appears.
7. Depending on the type of component you want to add to the update package, select one of the following from the list of components, and use the steps in the associated topics. You can add as many components as the update installation requires.
    1. Add a configuration file to an update package
    2. Add a file to an update package
    3. Add an installer file to an update package
    4. Add a patch file to an update package
    5. Add a script to an update package
8. Optional. Modify components as needed.
9. Click **OK**. PEDS Management Console is now validating the update package. Depending on the package size, it may take several minutes for the package to build. After the update package is generated, the PEDS Management Console screen appears.

    Note If a message appears stating that a package directory already exists, a package with the same revision or version number already exists. You must change either the package revision or version number.

Import an update package

An update package is an XML file that contains the installation files and the installer attributes you use to update an application. You can request an update package from Perceptive Software. To import an update package that is provided to you, complete the following steps.

1. Open the **PEDS Management Console** and select the **Package Generator** tab.
2. On the **Package Generator** tab, in the **Application** list, choose the application that is being updated.
3. To the right of the **Configure the package library** field, click **Import**. The **Import Packages** screen appears.
4. In the **Import Packages** screen, click **Browse** and navigate to the location of the deployment folder. To determine the deployment folder location, open the pedserver.ini file, and in the [General] section, note the value of the update.directory setting.

5. Select the location of the deployment folder and click **OK**. The package from the selected folder appears in the package area.
6. Select the package in the package area and click **OK**. Depending on the size of the package, it may take several minutes for the package to import. When the import is complete, the **Import Packages** screen closes and the **Package Generator** tab regains focus.

Modify an update package component

Update packages contain installer files, configuration files, scripts, and setup files. You can make changes to components of update packages. To modify a component of an existing update package, complete the following steps.

1. To modify an existing package component, in the **Configure the package library** field, select the package and click **Modify**.
2. In the **Modify Package** dialog box, use the steps in the topics below to make the appropriate changes to the update package.
   - Add a configuration file to an update package
   - Add a file to an update package
   - Add an installer file to an update package
   - Add a patch file to an update package
   - Add a script to an update package
3. Optional. To change the order of the components, select a component and click **Move Up** or **Move Down**.
4. Optional. To remove a component, select the component and click **Delete**.
5. Click **OK**. PEDS Management Console is now validating the update package. Depending on the package size, it may take several minutes for the package to build. After the update package is generated, the PEDS Management Console screen appears.

   Note If a message appears stating that a package directory already exists, a package with the same revision or version number already exists. You must change either the package revision or version number.

Add a configuration file to an update package

A configuration file contains application settings that are used during the installation of an update package. Configuration files are found inside update packages. To add a configuration file to an update package, complete the following steps.

1. On the **New Package** dialog box, click **Add** > **Configuration**.
2. In the **Configuration Setup** dialog box, click **Browse** and navigate to the location of the configuration file, and then click **Open**.
3. To add the install location for the configuration file, under the **Destinations** field, click **+** (plus sign).
4. In the **Destinations** field, keep **<installLocation>\,** and then type the subdirectory where the file will be installed. For example, to copy the file to the install location in a subdirectory of **inf**, enter the following string:

   ```
   <installLocation>\inf\
   ```

5. Optional. To remove a destination, select the location in the **Destinations** field and click **–** (minus sign).
6. Click **OK**. The **New Package** dialog box regains focus.

Add a file to an update package

Files of any type can be added to an update package. To add a file to an update package, complete the following steps.

1. In the **New Package** dialog box, click **Add** > **File**.
2. In the **File Setup** dialog box, click **Browse** and navigate to the location of the setup file, and then click **Open**.
3. To add the install location for the setup file, below the **Destinations** field, click **+** (plus sign).
4. In the **Destinations** field, keep **<installLocation>,** add a backslash, and then type the subdirectory where the file will be installed. For example, to copy the file to the install location in a subdirectory of inf, type the following string:

   ```
   <installLocation>\inf\
   ```
5. Optional. To remove a destination, select the location in the **Destinations** field and click **–** (minus sign).
6. Click **OK**.

Add an installer file to an update package

Installer files contain command line parameters and values that are used to execute certain commands during an installation. Installer files are found inside update packages. To add an installer file to an update package, complete the following steps.

1. In the **New Package** dialog box, click **Add** > **Installer**.
2. In the **Installer Setup** dialog box, click **Browse** and navigate to the location of the file, and then click **Open**.

   Note This can only be an EXE or MSI file.
3. Optional. Click **Advanced Options** to expand the **Installer Setup** dialog box.
4. In the **Arguments** field, enter a valid argument associated with the MSI file. For example, the following argument stops the installer from rebooting the computer after an installation is finished:

   ```
   REBOOT=ReallySupress
   ```
   - At the bottom of the **Properties** pane, click **+** (plus sign) and enter the **Name** and **Value** of a property associated with the installer file. Repeat this step to add additional variables.
   - At the bottom of the **Environment Variables** pane, click **+** (plus sign) and enter the **Name** and **Value** of an environment variable associated with the installer file. Repeat this step to add additional variables.
5. Click **OK**.

Add a patch file to an update package

Patch files contain only the updates that need to be applied to an application, instead of running an entire installer file to update an application. To add a patch file to an update package, complete the following steps.

1. In the **New Package** dialog box, click **Add** > **Patch**.
2. In the **Patch Setup** dialog box, click **Browse** and navigate to the location of the patch file, and then click **Open**.

   Note This can only be an EXE or MSP file.
3. Optional. Click **Advanced Options** to expand the **Patch Setup** dialog box.
4. In the **Arguments** field, enter a valid argument associated with the MSP file. For example, the following argument stops the installer from rebooting the computer after an installation is finished:

   ```
   REBOOT=ReallySupress
   ```

   - At the bottom of the **Properties** pane, click **+** (plus sign) and enter the **Name** and **Value** of a property associated with the patch file. Repeat this step to add additional variables.
   - At the bottom of the **Environment Variables** pane, click **+** (plus sign) and enter the **Name** and **Value** of an environment variable associated with the patch file. Repeat this step to add additional variables.
5. Click **OK**.

Add a script to an update package

A script is used to execute certain commands during installation of an update package. Scripts are found inside update packages. To add a script to an update package, complete the following steps.

1. On the **New Package** dialog box, click **Add** > **Script**.
2. In the **Script Setup** dialog box, do the following substeps:
   1. Click **Browse**, navigate to the location of the script, and then click **Open**.
   2. Optional. Click **Advanced Options** to expand the **Script Setup** dialog box.
3. In the **Arguments** field, enter a valid argument that you want to be passed into the script. For example, the following argument stops the installer from rebooting the computer after an installation is finished:

   ```
   REBOOT=ReallySupress
   ```

   - At the bottom of the **Environment Variables** pane, click **+** (plus sign) and enter the **Name** and **Value** of an environment variable associated with the script. Repeat this step to add additional variables.
4. Click **OK**.

What is a deployment plan?

A deployment plan is an XML file that specifies how and when to install an update package on users' computers. Each application that you are updating requires its own deployment plan.

The required elements of a deployment plan are the application identifier and the version that you are updating, the PEDS Server location, deployment type (including whether the update will be deployed As available or Forced), scheduling parameters, and the trigger type.

What are trigger types?

A trigger type is a component of a deployment plan that designates the method for initiating an installation. The following choices are available:

- **At application startup**. Updates are initiated only when the user tries to start the application. If an update is detected, it will be installed before the application is allowed to start.
- **Run in the background**. Updates are initiated on a periodic basis and run in the background. The application cannot be used while the update is being performed. The update type setting determines if the update will be performed As available or Forced.
- **Both**. Updates can be initiated when a user tries to start the application or they can run in the background.

What are update types?

Update packages can be deployed as either As available or Forced updates. The PEDS Notifier displays different messages depending on the update type.

- **As available**. A notification appears to inform the user that an update is available. The notification fades after ten seconds if there is no user action, or if the user minimizes it to the system tray. This action is repeated until the user closes the application and the update begins. The user can show the alert by selecting the icon in the system tray.
- **Forced update**. A dialog box appears on the user's computer and stays on top of any open applications. It contains a timer with the amount of time that is left before the system closes the application that is going to be updated. (Your administrator sets the amount of time to wait before performing this action in the deployment plan.) This dialog box also informs the user that an update is available and that the user needs to close open applications.

  Note If you use a forced update for multiple users on the same computer, all of the notification times are synchronized across that computer.

What are deployment types?

Deployment types determine how an update package is distributed to users' computers. The deployment types available with PEDS are installed and staged.

- **Installed**. The installed deployment type has the user's computer pull the update from the PEDS Server as soon as it is assembled and validated. The package resides on the user's computer until it is deployed either as an **As available** or **Forced update**.
- **Staged**. The staged deployment type has the user's computer pull the update from the PEDS Server as soon as it is assembled and validated. The update package resides on a user's computer until the schedule parameters in the deployment plan are met. When the schedule parameters are met, the update package is deployed either as an **As available** or **Forced update**.

Create a deployment plan

A deployment plan specifies when and how to install an update package on users' computers. The plan is specific to each application that you are upgrading. To create a deployment plan, complete the following steps.

Note Both installed deployment and staged deployment types are configured on the same tab of the PEDS Management Console. The Installed deployment type configuration is required. If you are only using an Installed deployment type in your deployment plan, leave the Staged deployment section empty or select None.

1. In the **PEDS Management Console**, on the **Deployment Plan** tab, in the **Application** list, select the application that you want to update.
2. In the **Server location** field, click **Browse** and select the location of your update directory.

   Example `C:\Program Files\apache-tomcat-7.0.53\webapps\pedserver\deployment`

   Note The update directory is configurable using the **update.directory** setting in the **General** section of the **pedserver.ini** file.
3. In the **Installed deployment** field, select the update package you want to install as soon as the users' computer checks for updates, based on the schedule.
4. In the **Schedule the deployment** area, select the update type **As available** or **Forced update**.

   Note The update type is only referenced when an update detects that the user is currently running the application.
5. Optional. If you selected **Forced update**, in the **Hour** and **Minutes** lists, select the time displayed to the user before the update begins.
6. Optional. To allow the reinstallation of a previous version of the application, select **Allow replacement of newer version**.
7. In the **Staged deployment** field, select the update package to use for the deployment.
8. In the **Schedule the deployment** area, select the update type **As available** or **Forced update**.

   Note The update type is only referenced when an update detects that the user is currently running the application.
9. Optional. If you selected **Forced update**, select the hours and minutes in the **Hour** and **Minutes** lists. This displays a message to the user, if the application is running, that shows a countdown to when the application will be closed and the update installed.
10. Optional. To allow the reinstallation of a previous version of the application, select the **Allow replacement of newer version** check box.
11. Optional. To set the staged deployment to an installed deployment on a certain date, select **Install the package on** check box, and in the **Select a date** and **Select a time** lists, select the date and time for the deployment.
12. In the **Trigger Type** area, select a method to initiate the installation.
13. Click **Publish**. Package validation begins and can take several minutes depending on the size of the update package.

About validating an update package

Validating update packages verifies the correctness of an update package without requiring you to publish it to the server.

Validating an update package occurs automatically when you finish creating or modifying it using PEDS Management Console.

Validating verifies the following:

- The packages specified in the deployment plan are valid packages.
- Confirms that the staging version is greater than the install version inside of the deployment plan.
- The revision number specified in the deployment plan for the install and staged package matches the revision of the package descriptor in each package.

About publishing an update package

An update package is published based on the parameters of its deployment plan.

The deployment type and trigger type settings configured in the deployment plan specify how and when an update package is published. If the update package is created using the PEDS Management Console, validation occurs when you click the OK button. You can also verify the update package manually using the command-line interface.

PEDS Launcher configuration settings

The following commands identify the application to update within the PEDLauncher.exe file.

Setting	Type	Description
appid	string	The application ID of the target.
application	string	The target application.

What are PEDS reports?

PEDS reports are web-based and provide detailed system information about ImageNow, Interact Desktop, and PEDS.

This information includes the current version, IP addresses, version is currently installed on the client, and updates that are staged for each computer in your system.

The landing page that contains the reports can be found at the following URL using a web browser:

<*server*>:8080/pedserver/Reports

Where <*server*> is the name of the PEDS server, and 8080 is the port number. Adjust the port number to the port used in your environment. This URL is case-sensitive.

Notes

- If you navigate to this URL on a child server, you are redirected to the parent server.

- Depending on the language version of Windows you are using, the date format that appears in the reports may not be localized.

Client Product Version Report - ImageNow

The Client Product Version Report - ImageNow in Perceptive Enterprise Deployment Reports contains information about the ImageNow Client or Perceptive Content Client on each client machine. The report contains the following information.

Column heading	Description	Example
Computer	This is the name of the computer on which ImageNow Client resides.	usercomputer1.cgi.com
IP	This is the IP address for the computer that ImageNow Client resides on.	123.45.600.78
Last Checked For Updates	This is the last date and time that PEDS checked the server for a new update package. **Note** Depending on the language version of Windows you are using, the date format that appears in this field may not be localized.	2012-01-02 8:45:30.123 Where: • 2012 is the year • 01 is the month • 02 is the day • 8 is the hour • :45 is the minutes • :30 is the seconds • .123 is the milliseconds
Installed Version	This is the current version of ImageNow Client that is installed on the client machine.	6.7.1.456
Staged Version 1	This is the version number of ImageNow Client in the update package that is on the client machine waiting to be installed.	6.7.1.789
Staged Version 2	This is the version number of the ImageNow Client in a second update package that is on the client machine waiting to be installed.	6.7.2.10

Client Product Version Report – Interact Desktop

The Client Product Version Report - Interact Desktop in Perceptive Enterprise Deployment Reports contains information about Interact Desktop on client machines. The report contains the following information.

Column heading	Description	Example
Computer	This is the name of the computer that Interact Desktop resides on.	usercomputer1.cgi.com
IP	This is the IP address for the computer that Interact Desktop resides on.	123.45.600.78
Last Checked For Updates	This is the last date and time that PEDS checked the server for a new update package. **Note** Depending on the language version of Windows you are using, the date format that appears in this field may not be localized.	2012-01-02 8:45:30.123 Where: • 2012 is the year • 01 is the month • 02 is the day • 8 is the hour • :45 is the minutes • :30 is the seconds • .123 is the milliseconds
Installed Version	This is the current version of Interact Desktop that is installed on the client machine.	6.7.1.456
Staged Version 1	This is the version number of the Interact Desktop in the update package that is on the client machine waiting to be installed.	6.7.1.789
Staged Version 2	This is the version number of Interact Desktop in a second update package that is on the client machine waiting to be installed.	6.7.2.10

PEDS Version Report

The PEDS Version Report in Perceptive Enterprise Deployment Reports contains information about your version PEDS. It allows you to identify the version of PEDS running on each client machines and so on. The report contains the following information.

Column heading	Description	Example
Computer	This is the name of the client machine on which PEDS resides.	usercomputer1.cgi.com

Column heading	Description	Example
IP	This is the IP address for the client machine on which PEDS resides.	123.45.600.78
Updater Version	This is the current version of the PEDS updater that resides on the client machine.	1.0.1.123

Appendix A: Using a command line interface with PEDS

All of the functions of the PEDS Management Console can also be performed using a command-line interface.

Create an update package using commands

An update package is a set of files that include the installation files, updater files, and an XML file that contains the installation file names and the installer attributes you use to update an application. To configure the contents of an update using a command-line interface, complete the following steps.

1. Navigate to the directory where pedconsole.exe is located on your machine.
2. Type all the following commands including their respective arguments:
3. `–create_package –installer_file <path> -application_id <identifier> -package_version <version>`
4. **Note** In this command string, <*path*>, <*identifier*>, and <*version*> are arguments that are specific to your system and installation.
5. Optional. Additional package configuration arguments can be added to the end of this command string. Refer to “Update package commands” for a list of available commands used to create update packages.
6. Press ENTER.

Validate an update package using commands

Validating your update package confirms that the staging version is greater than the install version inside of the deployment plan, and that the revision number specified in the deployment plan for the install and staged package matches the revision of the package descriptor in each package. It also confirms that all of the installation file copies in the deployment plan are from either the install directory <installFileLocation> or the installed location <installLocation>. Before you publish your update package to the server, you can validate the installation files. To validate an update package using a command-line interface, complete the following steps.

1. Refer to the Update package commands topic for a list of all available commands used to validate update packages. Navigate to the directory where pedconsole.exe is located on your machine.
2. Type all the following commands including their respective arguments:

   ```
   –validate -application_id <identifier> -base_package_dir <path>
   ```

   Note In this command string, <*identifier*> and <*path*> are arguments that are specific to your system and installation.
3. Optional. Add any optional commands you would like to include in the update package validation to the end of this command string.
4. Press ENTER.

Create a deployment plan using commands

A deployment plan specifies when and how to install an update package on users’ computers. The plan is specific to each application that you are upgrading. To create a deployment plan using a command-line interface, complete the following steps.

5. Navigate to the directory where **pedconsole.exe** is located on your machine.

6. Type all the following commands including their respective arguments:

   ```
   -create_plan -installer_file <path> -application_id <identifier> -
   install_type <install type>
   ```

7. **Note** In this command string, <*path*>, <*identifier*>, and <*install type*> are arguments that are specific to your system and installation.
8. Optional. Additional plan configuration arguments can be added to the end of this command string. Refer to “Deployment plan commands” for a list of available commands used to create deployment plans.
9. Press ENTER.

Modify a deployment plan component using commands

A deployment plan specifies when and how to install an update package on users’ computers. The plan is specific to each application that you are upgrading. To check for updates, PEDS points to the server to find a new deployment plan. You can make changes to components of existing deployment plans. To modify a component of an existing deployment plan using a command-line interface, complete the following steps.

Notes

- Refer to the “Deployment plan commands” for a list of available commands used for deployment plans and their descriptions.
- You create the update package after you have configured the deployment plan.

1. Navigate to the directory where pedconsole.exe is located on your machine.
2. Type all the following commands including their respective arguments:

   ```
   -edit_plan -application_id <identifier>
   ```

   Note In this command string, <*identifier*> is an argument that is specific to your system and installation.
3. Optional. Additional plan configuration arguments can be added to the end of this command string. Refer to “Deployment plan commands” for a list of available commands used to create update packages.
4. Press ENTER.

Management Console commands

The Management Console commands can be used to perform management console functions using the command-line interface.

Command	Argument	Description
\-?	N/A	Displays Help information for all of the commands.
\-? <command>	N/A	Displays Help for a specific command.
-? <command>	N/A	Displays more detailed information for a specific command.

\-gui	N/A	Starts the Management Console.

Update package commands

Use the update package commands to create or edit an update package using the command-line interface.

Command	Argument	Description
-create_package	N/A	Generates the update package.
-edit_package	N/A	Initiates an edit of the update package.
-installer_file	Path	The path to the installer file (an MSI, BAT, or EXE file) that installs the application update.
-application_id	Identifier	The application identifier.
-package_version	Version	Version of the application that is installed.
-base_package_dir	Path	The path to the directory to create the deployment package.
-installer_file	Path	The path to the installer file (an MSI, BAT, or EXE file) that installs the update.
-installer_type	Can be any of the following: Installshield Batch MSI	The type of installer to override with the file extension detection.
-installer_order	Integer	The order of the installer in the installation process.
-environment_variable	Parameters	The environment variable to set for the installer file. There can be multiple environment_variable arguments.
-installer_params	Parameters	Parameters that are passed into the installer.
-installer_property	Properties	Properties that are passed into the installer. format -installer_property "PROP=VALUE" There can be multiple arguments for this command.
-updater_source_dir	Path	Directory that contains the update files that are copied to the install directory (this is recursive). You can include multiple instances of this command.

Command	Argument	Description
-updater_source_file	Path	Update file that is copied to the update directory. There can be multiple arguments of the updater_source_file.
-install_source_dir	Path	Directory that contains the installation file that is copied to the install directory (this is recursive). There can be multiple arguments of the install_source_dir.
-install_source_file	Path	The install file that is copied to the installation directory. There can be multiple arguments of the install_source_file.
-duplicate_file_action	Command	Action to take when attempting to add a duplicate file to either the installation or update directory. It can be one of the following: **overwrite**. Overwrites the existing file, an error Is not displayed. **ignore**. Keeps the existing file, an error Is not displayed. **error**. Keeps the existing file and an error is displayed.
-config_file_copy	Configuration file copy information	This is the path to the file, the list of destinations, and the install order. For example: C:\Program Files\ImageNow6\etc\inow.ini=<installLocation>/etc/inow.ini;<installLocation>/bin/inow.ini@3. When this parameter is used, it replaces the configuration file copy information that is contained in the package descriptor.
-install_file_copy	Installation file copy information	This parameter is used to replace the installation file copy information that was in the package descriptor. For example: <pathToFile>=<installDestination>;<installDestination>@InstallOrder C:\Program Files\ImageNow6\etc\inow.ini=<installLocation>/etc/inow.ini;<installLocation>/bin/inow.ini@3 When this parameter is used, it replaces the install file copy information that is in the package descriptor.
-initial_package_version	Version number	The version of the package to open for editing. The default is the specified package_version.

Command	Argument	Description
-initial_revision	Integer	The revision of the package to open for editing. The default is the specified revision.
-revision	Integer	The revision of the package.
-delete_unreferenced_files	Flag	This command deletes the files that are no longer referenced in the deployment plan.
-install_location	-install_location	-install_location

Deployment plan commands

Use the deployment plan commands to create or edit a deployment plan using the command-line interface.

Command	Argument	Description
-create_plan	N/A	Generates the deployment plan.
-edit_plan	N/A	Edits the deployment plan.
-installer_file	Path	The path to the installer file (an MSI, BAT, or EXE file) that installs the application update.
-application_id	Identifier	Identifies the application that is installed.
-install_type	Install type	Alert users that an update is ready. Can be one of the following options: timed waitforuser If set to timed, the update begins at a certain time, and a message is displayed that informs the user that an update is ready to be installed and the user needs to exit the application. If set to waitforuser, the update is installed when the user exits the application.
-install_time	Integer	The number (integer) of minutes to wait for a timed update before closing the application. **Note** This is required if install_type is timed.
-install_version	Version number	The product version to install, for example: 6.7.0.1
-package_version	Version	Version of the application that is installed.

Command	Argument	Description
-base_package_dir	Path	The path to the directory where the application_id folder is located when the deployment plan is saved.
-install_version	Version number	The number of the version to install, for example: 6.7.0.1
-staging_type	Staging	Can be one of the following options: timed waitforuser This alerts the user that an install is ready and that they need to exit ImageNow (for a staged update, this occurs after the install after date has passed). Required if any staging parameter is used.
-staging_version	Version number	The install version to stage, for example: 6.7.0.1 Required if any staging parameter is used.
-staging_time	Integer	The number of minutes to wait for a timed update before closing the application. Required if staging_type is timed.
-staging_install_after_ts	Date	The date in the previous format (Z instead of -HH:MM is a UTC time), for example: 2011-07-28T09:40:00-06:00 2011-07-28T15:40:00Z
-trigger	Type	The mechanism for initiating a check for updates. Supply one of the following types: periodic onlaunch any
-display_name	String	The name of the application, displayed to the user, being updated.
-update_url	URL	The update server that the client uses to check for updates. Entered as a URL.
-deployment_version	Version number, such as x.x.x.x, where x is a number.	Version number of the deployment plan file, for example: 6.7.0.1

Command	Argument	Description
-install_upgrade_path	Upgrade Path type	This is the path for an installed upgrade. It can be either of the following options: **all**. Forces the full installation if you are not on the current version even if you are on a newer version. **newer**. The upgrade is only installed if it is a newer version, if it is only a revision change config is updated.
-staging_upgrade_path	Upgrade Path type	This is the path for a staged update. It can be either of the following options: **all**. Forces the full installation if you are not on the current version even if you are on a newer version. **newer**. The upgrade is only installed if it is a newer version, if it is only a revision change config is updated.
-install_revision	Integer	The revision number of the update package that is being deployed using an installed deployment.
-staging_revision	Integer	The revision number of the update package that is being deployed using a staged deployment.

PEDS Notifier commands

Use the PEDS Notifier commands to configure the notifier using the command-line interface.

Command	Argument	Description
-application_id	Identifier	The application ID that is used by the PEDS Client service to inform the notifier which updater to connect to.

PEDS Launcher

Use the PEDS Launcher commands to configure the launcher using the command-line interface.

Command	Argument	Description
-A or -application_id	Identifier	The application ID of the target.
-L or -application_exe	Path	The path to target executable.

Appendix B: XML file settings

Update package

An update package XML file is named <appId>.pedpkg, where <appId> is the application ID of the application you are updating. For example, the update package XML file for an ImageNow Client update for Windows 32-bit environments is ImageNow6Win32.pedpkg. The installLocation attribute on the updatePackage tag is used as the <installLocation>. It is only used if there is nothing in the registry for the install location. The revision attribute on the updatePackage tag is used to determine the revision number for the package.

Syntax

```
<updatePackage installLocation="" revision="1">
```

Variables

The registry value overrides the installLocation setting in the package. After a successful update, the installLocation that was used for the update is saved to the registry.

Variable	Description
installLocation	This is only used if there is nothing in the registry for the install location. This is used as the <installLocation> dictionary variable. The registry value overrides the setting in the package. **Note** After a successful update, the installLocation that was used for the update is saved to the registry.
revision	This is used to determine the revision number for the package.

Refer to the attribute values sections below for information about the attributes and revision. To view a detailed list of supported properties for the application(s) that you are updating, refer to the table in the “Run the unattended installation” section of the *Perceptive Content Client Installation Guide*.

```
<updatePackage installLocation="" revision="1">
  <installers>
    <installer type="InstallShield" installOrder="0">
      <executable path="install\ClientSetup_123" size="0" />
      <commandLineParameters>/qn REBOOT=ReallySuppress</commandLineParameters>
      <properties>
        <property name="IN_PORT_NO" value="6000" />
      </properties>
    </installer>
  </installers>
</updatePackage>
```

Installer attribute values

The <installers> array contains the <installer> attribute values found in the <appID>.pedpkg file.

Installer

```
<installer type="InstallShield" installOrder="0">
```

The installer choices are InstallShield, an MSI file, or a batch script. InstallShield and MSI indicate that the path in the executable tag points to an InstallShield or MSI (respectively). Batch indicates that the path points to a Windows batch script. The installOrder allows all of the installers and files to be ordered together, so that you can have files copied and have the installer run in any order.

Executable

```
<executable path="installClientSetup_73.exe" size="0" />
```

The path needs to be from either one of the paths in the dictionary; otherwise, it assumes the path is based off the installation directory (for example, <installFileLocation>\install.exe).

Command line parameters

```
<commandLineParameters>/qn REBOOT=ReallySuppress </commandLineParameters>
```

These are the command line parameters to pass into an MSI file.

```
/qn REBOOT=ReallySupress.

<property name="INSTALLDIR" value="<installLocation>" />
```

File attribute values

The <configurationFiles> and <files> arrays contain the <file> attribute values found in the <appID>.pedpkg file. All of the following attributes and tags are the same and perform the same functions. The difference between them arises when the -config_only argument is passed to the Updater, only the configurationFile section is processed. If the -config_only argument is omitted, all of the sections are processed.

```
<file installOrder="2">
```

This attribute is used to order the copying of files and the order the installers are run.

```
<source path="<installFileLocation>/somelocation/Test.xml" />
```

This attribute is the path portion of the attribute and needs to be one of the paths in the dictionary, for example <installFileLocation>\install.exe. Otherwise, it assumes the path is based off the install directory.

```
<destination useSourcePath="true" location=""<installLocation>\" />
```

useSourcePath element

The useSourcePath attribute flags whether it uses the source element's path attribute to complete the location of the destination. The location is either the base of the source element's path. The location attribute can use the directory dictionary to start the path, for example <installLocation>\config.ini.

The following is an example of the useSourcePath element. For the first destination, it is copied to <installLocation>\someLocation\Test.xml. The second destination is copied to <installLocation>\etc\Test3.xml. The third destination is copied to <installLocation>\etc\Test.xml. If the third destination was missing the slash at the end, it would attempt to rename the file to a file named “etc”.

```
<file installOrder="2">
  <source path="someLocation\Test.xml" />
  <destinations>
    <destination useSourcePath="true" location=" <installLocation>\" />
    <destination useSourcePath="false" location=" <installLocation>\etc\Test3.xml" />
    <destination useSourcePath="false" location=" <installLocation>\etc\" />
  </destinations>
</file>
```

Deployment plan

The deployment plan file is always named deployment_plan.xml. It describes aspects of the deployment that are independent of the actual update package. The following is an example deployment_plan.xml file.

```
<deploymentPlan version="0.0.0.0">
  <applicationDisplayName>ImageNow 6</applicationDisplayName>
  <updateURL>http://localhost:8080/pedserver</updateURL>
  <updateTriggerType>OnLaunch</updateTriggerType>
  <currentDeployment>
    <installUpdate upgradePath=”Newer”>
      <applicationVersion>6.7.0.45</applicationVersion>
      <packageRevision>0</packageRevision>
      <delayType>WaitForUser</delayType>
      <delayTime>1</delayTime>
    </installUpdate>
    <stageUpdate upgradePath=”Newer” installAfter="2011-06-28T14:40:00-05:00">
      <applicationVersion>6.7.0.46</applicationVersion>
      <packageRevision>0</packageRevision>
      <delayType>Timed</delayType>
      <delayTime>10</delayTime>
    </stageUpdate>
  </currentDeployment>
</deploymentPlan>
```

Deployment_plan.xml settings

The following table contains XML file settings used to configure a deployment plan.

Settings	Description
deploymentPlan version	Specifies the deployment version. **Note** This does not have to correspond to the version of the application. Currently unused.
appDisplayName	A user-friendly name for the application.
updateURL (Optional)	The location of the server to check for updates on.
updateTriggerType	Indicates when the check for updates is performed. • **Periodic**. A check is done by the service at fixed intervals. • **OnLaunch**. A check is done when the user starts the application -using the PEDS Launcher. • **Any**. A combination of Periodic and OnLaunch.
currentDeployment	The section that describes the two deployment types, either installed or staged.

Settings for an installed update

The following table contains XML file settings used to configure an installed update.

Setting	Description
upgradePath	This can be either of the following options: **Newer**. Specifies that the update is installed only if it is newer than the currently installed version. **Note** Configuration files are installed if the update version is the same as the currently installed version. **All**. Specifies that the update is installed regardless if it is newer or older than the currently installed version.
applicationVersion	Version of the application to download and install.
packageRevision	Revision of the package that is installed. If upgradePath is Newer, only the configuration files (indicated in the package descriptor) are installed. If upgradePath is All, then the entire package is installed.
delayType	This can be either of the following options: **Timed**. The user is forced out of the application if it is not closed within the configured delayTime. **WaitForUser**. The update waits for the user to close the application.

delayTime	When delayType equals Timed, this setting indicates how long (in minutes) the user will be given to close the application.

Settings for a staged update

The following table contains XML file settings used to configure a staged update.

Setting	Description
installAfter (Optional)	The date on which the stageUpdate becomes the installUpdate.
upgradePath	This can be either of the following options: **Newer**. Specifies that the update is installed only if it is newer than the currently installed version. **Note** configuration files are installed if the update version is the same as the currently installed version. **All**. Specifies that the update is installed regardless if it is newer or older than the currently installed version.
applicationVersion	Version of the application to stage (to download but not install).
packageRevision	Revision of the package that is installed. This number is used to determine whether or not to update when the update version is the same as the currently installed version. If upgradePath is Newer, only the configuration files (indicated in the package descriptor) are installed. If upgradePath is All, then the entire package is installed.
delayType	This can be either of the following options: **Timed**. The user is forced out of the application if it is not closed within the configured delayTime. **WaitForUser**. The update waits for the user to close the application.
delayTime	When delayType equals Timed, this setting indicates how long (in minutes) the user will be given to close the application.