

LISTEN.  
THINK.  
SOLVE.®

# PharmaSuite®



## INSTALLATION - UPGRADE RELEASE 8.4 TECHNICAL MANUAL

PUBLICATION PSUP-IN005E-EN-E-DECEMBER-2017  
Supersedes publication PSUP-IN005D-EN-E



*Allen-Bradley • Rockwell Software*

**Rockwell  
Automation**



**Contact Rockwell** See contact information provided in your maintenance contract.

**Copyright Notice** © 2017 Rockwell Automation Technologies, Inc. All rights reserved.  
This document and any accompanying Rockwell Software products are copyrighted by Rockwell Automation Technologies, Inc. Any reproduction and/or distribution without prior written consent from Rockwell Automation Technologies, Inc. is strictly prohibited. Please refer to the license agreement for details.

**Trademark Notices** FactoryTalk, PharmaSuite, Rockwell Automation, Rockwell Software, and the Rockwell Software logo are registered trademarks of Rockwell Automation, Inc.

The following logos and products are trademarks of Rockwell Automation, Inc.:

FactoryTalk Shop Operations Server, FactoryTalk ProductionCentre, FactoryTalk Administration Console, FactoryTalk Automation Platform, and FactoryTalk Security.

Operational Data Store, ODS, Plant Operations, Process Designer, Shop Operations, Rockwell Software CPGSuite, and Rockwell Software AutoSuite.

**Other Trademarks** ActiveX, Microsoft, Microsoft Access, SQL Server, Visual Basic, Visual C++, Visual SourceSafe, Windows, Windows 7 Professional, Windows Server 2008, Windows Server 2012, and Windows Server 2016 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Adobe, Acrobat, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

ControlNet is a registered trademark of ControlNet International.

DeviceNet is a trademark of the Open DeviceNet Vendor Association, Inc. (ODVA).

Ethernet is a registered trademark of Digital Equipment Corporation, Intel, and Xerox Corporation.

OLE for Process Control (OPC) is a registered trademark of the OPC Foundation.

Oracle, SQL\*Net, and SQL\*Plus are registered trademarks of Oracle Corporation.

All other trademarks are the property of their respective holders and are hereby acknowledged.

**Warranty** This product is warranted in accordance with the product license. The product's performance may be affected by system configuration, the application being performed, operator control, maintenance, and other related factors. Rockwell Automation is not responsible for these intervening factors. The instructions in this document do not cover all the details or variations in the equipment, procedure, or process described, nor do they provide directions for meeting every possible contingency during installation, operation, or maintenance. This product's implementation may vary among users.

This document is current as of the time of release of the product; however, the accompanying software may have changed since the release. Rockwell Automation, Inc. reserves the right to change any information contained in this document or the software at any time without prior notice. It is your responsibility to obtain the most current information available from Rockwell when installing or using this product.



<b>Chapter 1</b>	<b>Introduction .....</b>	<b>1</b>
	Intended Audience .....	1
	Typographical Conventions .....	2
<b>Chapter 2</b>	<b>Upgrading an Installation .....</b>	<b>3</b>
	Organizational Process for the Rollout of an Upgrade .....	5
	Performing the Upgrade .....	6
	Updating the Platform .....	6
	Updating FactoryTalk ProductionCentre .....	8
	Updating the Audit Trail Configuration.....	9
	Updating the System .....	11
	Running the Update Wizard .....	15
	Migrating the Data.....	25
	Running the Data Migration Wizard .....	27
	Installing MR+ Revisions of Building Blocks .....	36
	Removing Duplicate Classes from the Classpath .....	37
	Reporting Upgrade Results .....	38
	Logging the Upgrade .....	42
	Automated Installation of the Upgrade .....	43
	Automated Wizard Installation .....	43
	Console Mode Installation .....	45
<b>Chapter 3</b>	<b>Technical Details of a System Update .....</b>	<b>47</b>
	General Update Strategy .....	47
	How to Resolve Conflicts .....	49
	Update Tasks .....	50

Checking for Checked-out Objects.....	51
Updating UDA Definitions.....	51
Updating Library Objects.....	53
Updating DSX Objects (Generic DSX) .....	53
Updating AT Definitions .....	57
Updating ATRow Objects .....	59
Updating Access Privilege Objects .....	61
Updating Image Objects .....	61
Updating FSM Objects.....	62
Updating Message Objects .....	67
Updating Application Objects.....	68
Updating Versioning Configuration Objects.....	68
Updating Report Design Objects.....	69
Updating EAR Files .....	70
Reporting Customer Modifications .....	71
<b>Chapter 4 Technical Details of a Data Migration.....</b>	<b>73</b>
Blockwise Update .....	73
Migration Tasks .....	74
Migrating Transition Conditions of Activity Sets .....	75
Migrating Usage List-related Tables .....	76
<b>Chapter 5 Adapting the Upgrade Installer .....</b>	<b>77</b>
Detecting Artifacts that Need to Be Adapted .....	78
Resolving DSX Conflicts.....	78
Merging DSX Artifacts .....	79
Creating New Upgrade Tasks .....	80
Interface to Be Implemented .....	82
Useful Base Classes.....	83
Relevant APIs.....	83
Writing Unit Tests for Tasks.....	85
Creating a JAR from Eclipse.....	85

Guidelines for Implementing Data Migration Tasks.....	85
Processing a List of TaskResultItems vs. Logging .....	86
Loading Objects to Be Migrated .....	86
Updating the Target System .....	87
Verifying the Updated System .....	87
Example of a Data Migration Task.....	87
<b>Chapter 6   Reference Documents .....</b>	<b>91</b>
<b>Chapter 7   Revision History .....</b>	<b>93</b>
<b>Index .....</b>	<b>97</b>





Figure 1: Operating principles of the upgrade engine .....	4
Figure 2: Rollout of an upgrade .....	5
Figure 3: Command prompt with startup parameter .....	15
Figure 4: Welcome .....	16
Figure 5: Extraction Directory .....	16
Figure 6: Extraction Packages .....	17
Figure 7: Extraction .....	17
Figure 8: Update Settings.....	18
Figure 9: Checking Update Configuration.....	19
Figure 10: Computing Planned Changes.....	20
Figure 11: Checking Target System for Conflicts - in progress.....	21
Figure 12: Checking Target System for Conflicts - completed .....	21
Figure 13: Performing Update on Target System .....	22
Figure 14: Verifying Updated Target System - in progress.....	23
Figure 15: Verifying Updated Target System - completed .....	23
Figure 16: Update Complete .....	24
Figure 17: Command prompt with startup parameter .....	27
Figure 18: Welcome .....	28
Figure 19: Extraction Directory.....	28
Figure 20: Extraction Packages .....	29
Figure 21: Extraction .....	29
Figure 22: Data Migration Settings .....	30
Figure 23: Checking Configuration .....	31
Figure 24: Computing Planned Changes.....	32
Figure 25: Checking Target System for Conflicts .....	33
Figure 26: Migrating Data on Target System .....	34
Figure 27: Verifying Migrated Target System .....	35

Figure 28: Data Migration Complete .....	35
Figure 29: Example of a duplicate library .....	38

## Introduction

This manual describes the procedure for upgrading a configured and extended PharmaSuite system with a new standard system while retaining its configurations and extensions.

### Intended Audience

The manual is intended for system integrators and administrators of a PharmaSuite system planning to update and migrate PharmaSuite and to adapt the upgrade installer.

## Typographical Conventions

This documentation uses typographical conventions to enhance the readability of the information it presents. The following kinds of formatting indicate specific information:

<b>Bold typeface</b>	Designates user interface texts, such as <ul style="list-style-type: none"><li>■ window and dialog titles</li><li>■ menu functions</li><li>■ panel, tab, and button names</li><li>■ box labels</li><li>■ object properties and their values (e.g. status).</li></ul>
<i>Italic typeface</i>	Designates technical background information, such as <ul style="list-style-type: none"><li>■ path, folder, and file names</li><li>■ methods</li><li>■ classes.</li></ul>
CAPITALS	Designate keyboard-related information, such as <ul style="list-style-type: none"><li>■ key names</li><li>■ keyboard shortcuts.</li></ul>
Monospaced typeface	Designates code examples.

### TIP

Instructions in this manual are based on Windows 7. Select the appropriate commands if you are using a different operating system.

## Upgrading an Installation

This section contains general information about upgrading an existing PharmaSuite installation.

Upgrading PharmaSuite involves the following sequence of activities:

1. Updating the underlying FactoryTalk ProductionCentre system (only necessary if the PharmaSuite system to which you are about to upgrade is based on a different platform version/build).
2. Updating the PharmaSuite system to the new software version by applying functional and structural changes and adding new functions.
3. Migrating the data that resides in the newly updated PharmaSuite system, especially master recipes and custom building blocks, which are assessed and adjusted to be suitable for use with the new or changed system functions.
4. Finally, it may be necessary to update the software of some phase building blocks for use with the newly upgraded PharmaSuite system.

The upgrade engine covers the update of the PharmaSuite system and the migration of the system data. It compares the **BASE** version of PharmaSuite, which is the old standard system, with the **NEW** standard version of the system to identify all differences. The resulting differences version **DIFF** is then merged with the **TARGET** version, which is the configured and possibly extended customer system, to obtain a **FINAL** customer version that includes the changes and new functions from the standard system as well as all customer-specific configurations and extensions.

Technically, the upgrade engine relies on the DSX files representation of the PharmaSuite system, generated by the DSX export capability of the FactoryTalk ProductionCentre platform. It performs all comparison and merge operations on the exported DSX files.

To perform the comparison and merge activities, the upgrade engine relies on the DSX export capability provided by the platform. Please be aware of the underlying concept that the whole update process relies on a DSX export aka DSX files representation of the system.

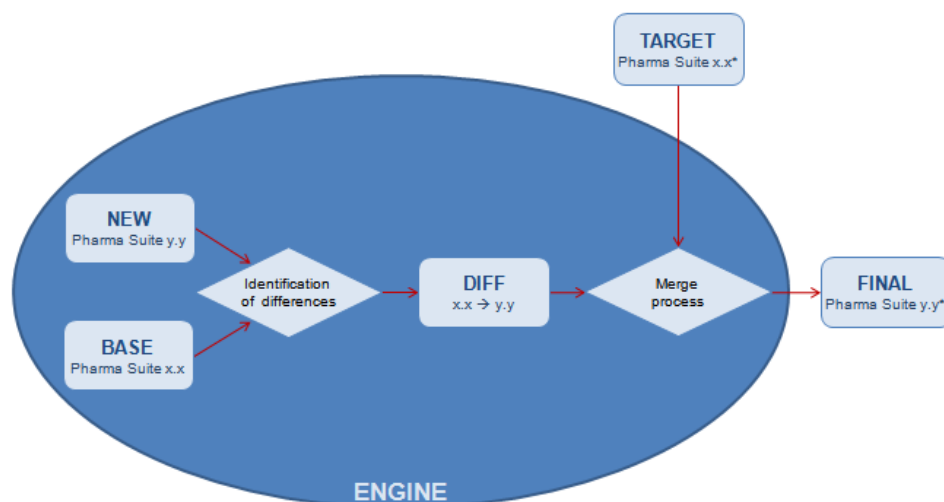


Figure 1: Operating principles of the upgrade engine

The engine performs one run for updating the system and a second run for migrating the data, with each run consisting of five stages. In each stage, the upgrade tasks of specific artifacts are executed. The stages are embedded in a wizard that prepares the upgrade process by copying the files and collecting the settings of the target application server. A stage can only be finished if each upgrade task has been completed successfully. For more information, see "Performing the Upgrade" (page 6) and "Reporting Upgrade Results" (page 38).

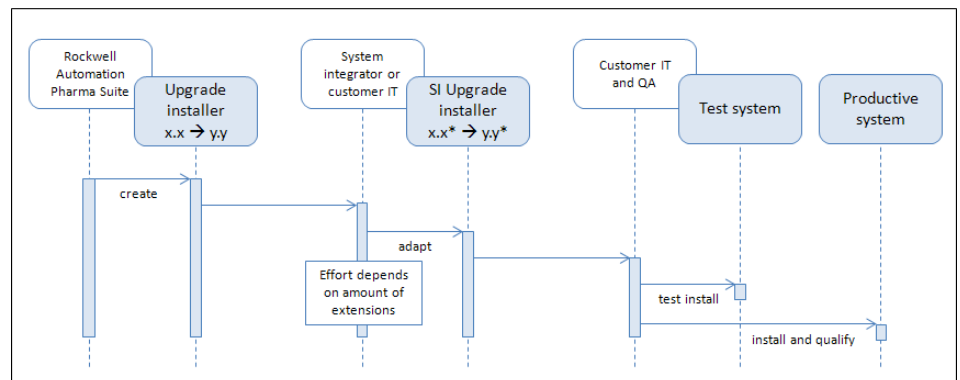
The five upgrade stages are:

- **Stage 1** (Checking Update Configuration)/(Checking Configuration)  
Checks the integrity of the upgrade installer. The stage evaluates whether each of the files located in the **BASE\_VERSION** and **NEW\_VERSION** folders of the respective upgrade directory is handled by exactly one upgrade task. Incomplete configurations are reported as warnings thus allowing you to exclude certain artifacts from the upgrade process.
- **Stage 2** (Computing Planned Changes)  
Computes which changes are needed to upgrade from the **BASE** version to the **NEW** version. This result list is independent of the actual contents of the target system.
- **Stage 3** (Checking Target System for Conflicts)  
Checks if any of the planned changes are in conflict with the content of the **TARGET** system. All conflicts will be reported as errors. Planned changes that have already been performed on the **TARGET** system are removed from the list of required changes.

- **Stage 4** (Performing Update on Target System)/(Migrating Data on Target System)  
Executes the required changes on the **TARGET** system.  
Issues are reported as errors, except for failed deletions, which are presented as warnings.
- **Stage 5** (Verifying Updated Target System)/(Verifying Migrated Target System)  
Verifies that all changes were successfully executed on the target system. The stage compares the expected version with **NEW** version.  
Changes that were planned but not performed are reported as errors, except for deletions, which may be skipped. For details, see the descriptions of the object-specific upgrade tasks.

## Organizational Process for the Rollout of an Upgrade

The organizational process for rolling out the upgrade of a PharmaSuite system can consist of up to three different process phases which usually also involve different roles and organizations.



*Figure 2: Rollout of an upgrade*

In the first phase, we deliver an upgrade installer package that updates a standard PharmaSuite installation and migrates existing data.

In the second phase, a system integrator or a customer's IT department run the upgrade on their configured and possibly extended customer system in a test environment. As a result of the test run, the system integrator receives an overview of potential or necessary adaptations caused by conflicts between the new standard version and the specific configurations and extensions added to the customer version. When the system integrator has addressed and resolved all conflicts, the outcome of this stage is a customer-specific version of the upgrade installer package.

The adaptation of the upgrade installer (page 77) can be achieved in an iterative approach, to resolve conflict by conflict or to cover site-specific changes of a customer.

If no adaptations are necessary, the original PharmaSuite upgrade installer can be used. Anyway, we strongly recommend to extensively test the upgrade installer package on a copy of the real customer system to prevent conflicts from occurring during the installation on a production environment.

In the final phase, the customer IT and QA perform a test installation of the upgrade installer. After a successful test, apply the upgrade to the productive system.

## Performing the Upgrade

Before you can begin to upgrade the PharmaSuite 8.3 system to PharmaSuite 8.4, you need to make sure that all software and hardware prerequisites are met. For details regarding the required server and client hardware, the operating system, and the database server, please refer to the "PharmaSuite Supported Platforms Guide" [A1] (page 91) and the "FactoryTalk ProductionCentre Release 10.4 Supported Platforms Guide" [A2] (page 91).

When you have confirmed the prerequisites you can proceed with updating the underlying FactoryTalk ProductionCentre platform system. Afterwards your system is ready for upgrading PharmaSuite. There are two separate wizards, one for updating the system, the other for migrating the data, which you need to run one after another in order to perform the complete upgrade. Both the update and the migration wizards perform the five upgrade stages (page 3) and provide dedicated stage result files. They can be performed user-driven or as automated installation.

## Updating the Platform

Please use the prerequisites checklist to verify that all preparatory steps have been completed and you have all information available you will need during the update process.

The following checklist covers all preparatory steps for updating a FactoryTalk ProductionCentre system.



### PREREQUISITES CHECKLIST

Before you start with the update of FactoryTalk ProductionCentre, check the prerequisites:

	Prerequisite	Your Notes	Done?
1	Adequate hardware as defined in "PharmaSuite Supported Platforms Guide" [A1] (page 91) and "FactoryTalk ProductionCentre Release 10.4 Supported Platforms Guide" [A2] (page 91).		
2	Operating system software installed. See "PharmaSuite Supported Platforms Guide" [A1] (page 91).		
3	Database server software installed. See "PharmaSuite Supported Platforms Guide" [A1] (page 91) and "FactoryTalk ProductionCentre Release 10.4 Database Installation Guide" [A3] (page 91).		
4	User who will perform the system update has Administrator rights to the operating system.		
5	There are no locked objects (see <b>Remove Object Locks</b> task of the Production Management Client).		
6	<i>PharmaSuite_Enterprise_&lt;version&gt;.zip</i> (included in <version>-PharmaSuite-FTPSJBOSS-DVD.zip) downloaded from the Rockwell Automation Download Site (page 91), saved locally or in a network directory, and unpacked. <version> stands for the final version and build number of PharmaSuite Update.		

## Updating FactoryTalk ProductionCentre

To update the FactoryTalk ProductionCentre platform and thus prepare the system for the PharmaSuite upgrade, proceed as follows:

### IMPORTANT

Before you start to update the platform, make sure to create a backup copy of the *PharmaSuite-Help.ear* file, which is located in the JBoss directory (`..\JBoss\jboss-eap-7.0.0\standalone\deployments\PharmaSuite-Help.ear`). You will later need to copy the file to the same location in the updated JBoss directory.

1. Update the FactoryTalk ProductionCentre system as described in "FactoryTalk ProductionCentre Plant Operations Release 10.4 Server Installation Guide - JBoss Advanced" [A6] (page 91). Refer to Chapter 1 "Installation Checklists", sections "Upgrade Single Application Server Checklist" or "Upgrade Clustered Application Server Checklist".
2. You need to uninstall all Shop Operations servers of the previous version of FactoryTalk ProductionCentre before you can install new instances for the current version of FactoryTalk ProductionCentre.

### TIP

Retain the *ShopOperationsServer.xml* configuration files of the old 10.3 Shop Operations servers, since they contain the settings to be used for your new 10.4 Shop Operations servers.

3. Copy the backup of the *PharmaSuite-Help.ear* to the updated JBoss directory (`..\JBoss\jboss-eap-7.0.0\standalone\deployments\PharmaSuite-Help.ear`).
4. Make sure, that your Shop Operations servers at set up according to section "Setting up Shop Operations Servers" in "Technical Manual Installation - Enterprise Edition" [A7] (page 91).

Proceed with updating the audit trail configuration (page 9).

## Updating the Audit Trail Configuration

Generally, PharmaSuite has audit trail enabled for all of its objects. To reduce database growth, however, some objects are excluded from this general rule. The set of objects that is excluded from audit trail can differ between PharmaSuite releases. Identifying the differences in the audit trail configuration and performing the corresponding updates to the database need to be performed manually before you can update the PharmaSuite system. Proceed as follows:

1. In the location where you have saved the extracted *PharmaSuite\_Enterprise\_<version>* directory before, navigate to the *..\conf\DisableAuditTrail.properties* file, which defines all database tables for which audit trail is disabled. Create a copy of the file in a temporary folder to work on.
  - To disable audit trail for additional objects, such as project-specific custom ATRows, add the respective database tables to your working copy of the *DisableAuditTrail.properties* file.

### TIP

Before you disable audit trail for an object, make sure that its audit information is really not needed.

- To enable audit trail for specific database tables, delete the respective entries from the *DisableAuditTrail.properties* file.

### TIP

When you enable audit trail for an object, you need to be aware that this affects database growth.

2. Compare the database tables listed in the *DisableAuditTrail.properties* file, representing the objects for which audit trail is to be disabled, with the tables listed in the XFR\_AUDIT\_OVERRIDE table of your system's database, representing the objects for which audit trail is currently disabled.
  - If a database table is listed in the *DisableAuditTrail.properties* file, but not in the XFR\_AUDIT\_OVERRIDE table:
    - Consider to disable audit trail for this table by adding it to XFR\_AUDIT\_OVERRIDE table, unless audit trail has been intentionally enabled for this table for project-specific purposes.
    - When you have decided to disable audit trail for a database table and have added it to the XFR\_AUDIT\_OVERRIDE table, also consider to remove the data from the respective audit table.
  - If a database table is listed in the XFR\_AUDIT\_OVERRIDE table, but not in the *DisableAuditTrail.properties* file:

- Consider to enable audit trail for this table by removing it from the XFR\_AUDIT\_OVERRIDE, unless audit trail has been intentionally disabled for this table for project-specific purposes.

**TIP**

Make sure not to enable audit trail for phase-related runtime tables that have been added to the XFR\_AUDIT\_OVERRIDE table intentionally. For further details, please refer to section "Configuring Audit Trail", chapter "Performing the Installation" in "Technical Manual Installation - Building Block" [A9] (page 91).

3. If you have modified the XFR\_AUDIT\_OVERRIDE table, execute the *dsResetAuditTriggers* stored procedure:
  - For Oracle databases:  
`declare begin dsResetAuditTriggers(); end;`
  - For SQL databases:  
`exec dsResetAuditTriggers`

Proceed with updating the system (page 11).

## Updating the System

Please use the prerequisites and information checklists to verify that all preparatory steps have been completed and you have all information available you will need during the update process.

The following two checklists cover all preparatory steps and information required for updating a PharmaSuite system.

### INFORMATION CHECKLIST

Please prepare the information you will need during the update process:

	Information	Your Notes	Done?
1	Application server: Remote URL		
2	Application server: HTTP URL		
3	Application server: Login ID (user name) with required privileges to PharmaSuite (is member of at least the <b>MinimalAccess</b> and the <b>PlantOpsDesigner</b> user groups)		
4	Application server: Password		
5	Existing system installation directory: EAR file deployment directory		
6	System configuration considers the following settings: ActiveMQ and the PharmaSuite upgrade engine run with the 64-bit version of Java 1.8.0_144. Each PharmaSuite client (including Shop Operations Server) is only supported for the 32-bit version of Java 1.8.0_144. Make sure that the specified programs use a JAVA_HOME environment variable pointing to the correct Java version.		

### TIPS

Please do not put your password in writing, only include a hint to it in your notes.

To configure which default values are preset for settings such as the application server information, navigate to the downloaded *update-package-builder-<version>.jar\resources* directory and open the *userInputSpec.xml* file in a text editor.

You can define a default by editing or defining the `set` attributes of the `spec txt` tags:  
`<spec txt="Remote URL" size="50" set="remote://MyServer:8080" />`

The following values can be preset with a default:

- Remote URL
- Server HTTP URL
- Password
- Confirm password
- EAR deployment directory

## PREREQUISITES CHECKLIST

Before you start the update wizard, check the prerequisites:

	Prerequisite	Your Notes	Done?
1	FactoryTalk ProductionCentre platform system updated. See "FactoryTalk ProductionCentre Plant Operations Release 10.4 Server Installation Guide - JBoss Advanced" [A6] (page 91).		
2	Audit trail configuration updated.		
3	User who will perform the system update has Administrator rights to the operating system.		
4	User whose credentials will be used to access the application server during the system update has the required privileges to PharmaSuite (is member of at least the <b>MinimalAccess</b> and the <b>PlantOpsDesigner</b> user groups) and references the <b>DefaultConfiguration</b> application.		
5	Services of all PharmaSuite-related event sheets stopped. See "Technical Manual Administration" [A4] (page 91).		
6	All data objects on the <b>TARGET</b> system are checked in with Process Designer and thus saved to the database.		
7	Backup copy of <i>PharmaSuite-Help.ear</i> file copied to updated JBoss directory (..\JBoss\jboss-eap-7.0.0\standalone\deployments\PharmaSuite-Help.ear).		
8	Update wizard ( <i>update-package-builder- &lt;version&gt;.jar</i> ) downloaded from the Rockwell Automation Download Site (page 91) and saved locally or in a network directory. <version> stands for the final version and build number of PharmaSuite Update.		

	Prerequisite	Your Notes	Done?
9	<p>Transaction timeout settings in JBoss configuration adapted.</p> <p>In the <code>&lt;JBoss_DIR&gt;\standalone\configuration\standalone-<i>full.xml</i></code> file, identify the <code>&lt;subsystem xmlns="urn:jboss:domain:transactions:3.0"&gt;</code> section and add the following line at the end of the section:</p> <pre>&lt;coordinator-environment default-timeout="600"/&gt;</pre> <p>Make sure that the JBoss Application Server is restarted to make the change take effect.</p>		



## Running the Update Wizard

To execute the system update, proceed as follows:

1. You need to run the update wizard with a startup parameter to make sure you have sufficient memory assigned to the process. In the directory where you have saved the *update-package-builder-<version>.jar* file before, open a command prompt as administrator and type as follows:

```
java -Xmx1024m -jar update-package-builder-<version>.jar
```

Then press the ENTER key to start the update wizard.

### TIP

Please note that the command prompt needs to remain open during the entire run of the wizard. You can minimize it to the system tray but must not close it.

You can improve the performance of the wizard by running it as close to the application server as possible, ideally on the same machine as the application server, since the process involves a large number of middle-tier calls.

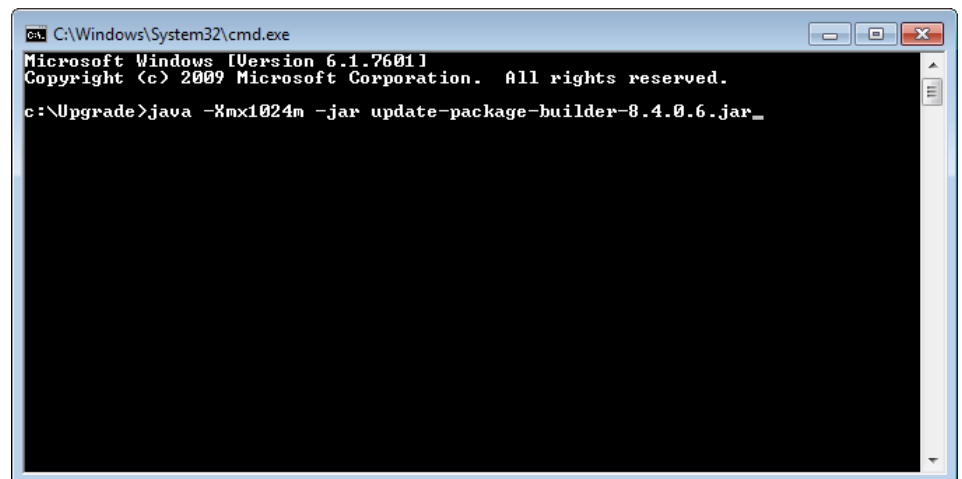


Figure 3: Command prompt with startup parameter

2. Wait until the welcome dialog opens. Click the **Next** button to continue.

**TIP**

Please note that the PharmaSuite version/build number displayed by the wizard you are running may be higher than the one shown on the screen captures in this manual.

For the exact build number valid for the PharmaSuite release you are about to install, refer to section "Released Product Information" of the "Quality Certificate" [A5] (page 91) of this release.

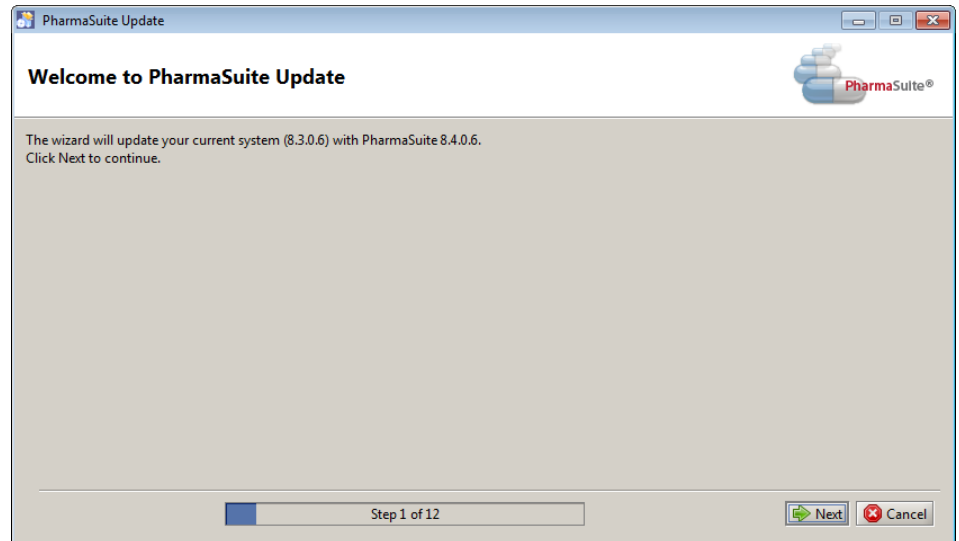


Figure 4: Welcome

3. Select the directory to which you wish to extract the update installer. To change the default path, click the **Browse** button and select another directory. Click the **Next** button to continue.

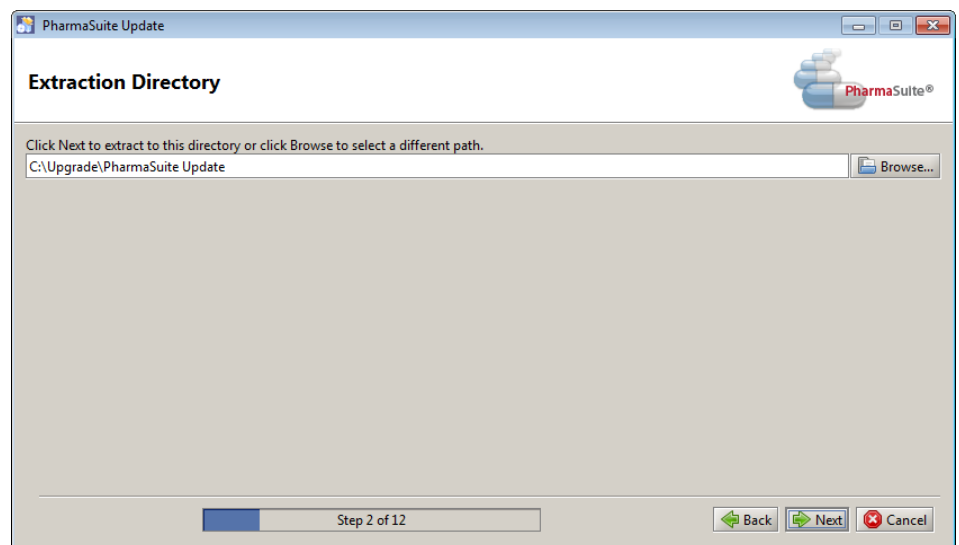


Figure 5: Extraction Directory

4. The system lists the packages that are available for extraction. The **Update Engine** and **Base and New Version** packages are required to run the update and thus cannot be unselected. You only need to extract the **SDK for the Update Package** if you plan to adapt the upgrade installer (page 77) itself and build a new version of the update wizard. Select the packages you wish to extract. Click the **Next** button to continue.

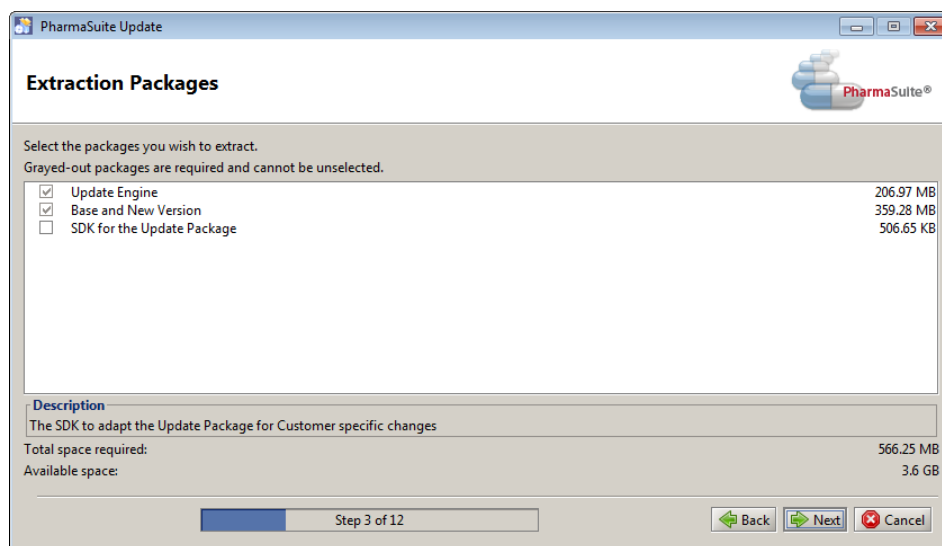


Figure 6: Extraction Packages

**TIP**

The selection status of the optional **SDK for the Update Package** package is not displayed correctly. When you select this package, the checkbox display only becomes properly visible when you select another row.

5. Wait until all packages have been extracted. Click the **Next** button to continue.

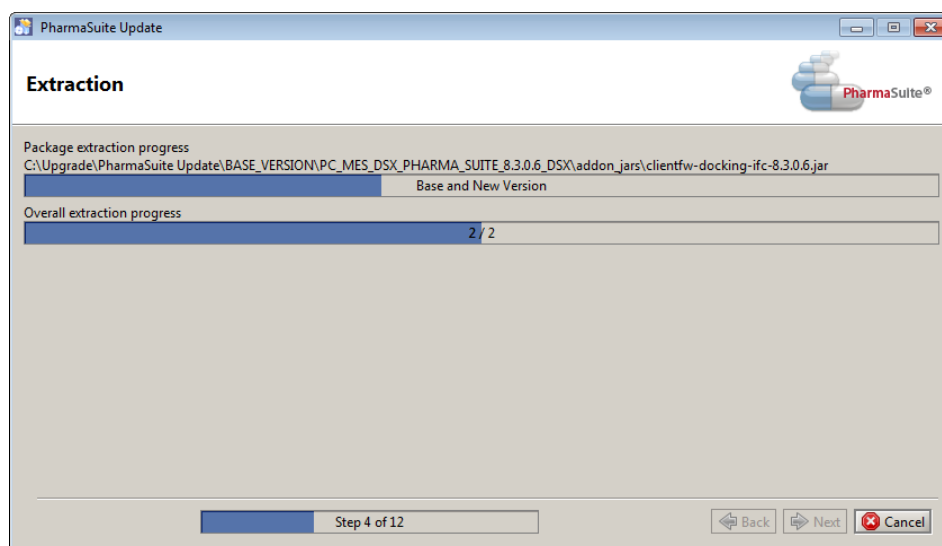


Figure 7: Extraction

6. Provide the data for the application server connection and the deployment path for the EAR file:
  - **Remote URL** of your application server (here: **remote://ftpsmigration04:8080**).
  - **Server HTTP URL** of your application server (here: **http://ftpsmigration04:8080**).
  - **User** (login name) and **Password** of a user with suitable access privileges to perform the update (must be member of at least **MinimalAccess** and **PlantOpsDesigner** user groups).
  - EAR deployment directory indicates the path to the PharmaSuite help EAR file in the system you are about to update.

**TIP**

Please note that the preset values in the input boxes can be edited. To change the presets themselves, you need to adjust the default values before you run the migration wizard. They are defined in the *userInputSpec.xml* file in the downloaded *update-package-data-<version>.jar\resources* directory.

Click the **Next** button to continue.

The screenshot shows a window titled 'PharmaSuite Update' with a sub-header 'Update Settings'. It contains two sections: 'Application server information' and 'System installation information'. The first section has five input fields: 'Remote URL' (remote://ftpsmigration04:8080), 'Server HTTP URL' (http://ftpsmigration04:8080), 'User' (pmcadmin), 'Password' (masked with dots), and 'Confirm password' (masked with dots). The second section has one input field: 'EAR deployment directory' (Q:\Boss\jboss-eap-7.0.0\standalone\deployments), followed by a 'Browse...' button. At the bottom, there is a progress bar labeled 'Step 5 of 12' and three buttons: 'Back', 'Next', and 'Cancel'.

Figure 8: Update Settings

**IMPORTANT**

If you have accidentally entered an incorrect URL in this step and clicked the **Next** button, the system informs you that it cannot connect to the application server. Please note that you cannot use the **Back** button to return to the step for correcting your input, but have to click the **Cancel** button to terminate the wizard and then restart it (with the startup parameters). However, if an erroneous input is located in the **User** or **Password** boxes, you can use the **Back** button to step back and correct your input.

7. The system starts stage 1 of the update process and checks the update configuration.

**TIPS**

Please note that you cannot cancel an update stage in mid-process. When you click the **Cancel** button, the system first completes all tasks of the stage to ensure data and log file consistency before it terminates the update process.

Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the update installer and is called *CHECK\_PACKAGE\_CONFIGURATION.xml*.

Click the **Next** button to continue.

**TIP**

Look at the individual result files (page 38) of the 5 stages to see if the respective tasks have identified issues with the objects they handle.

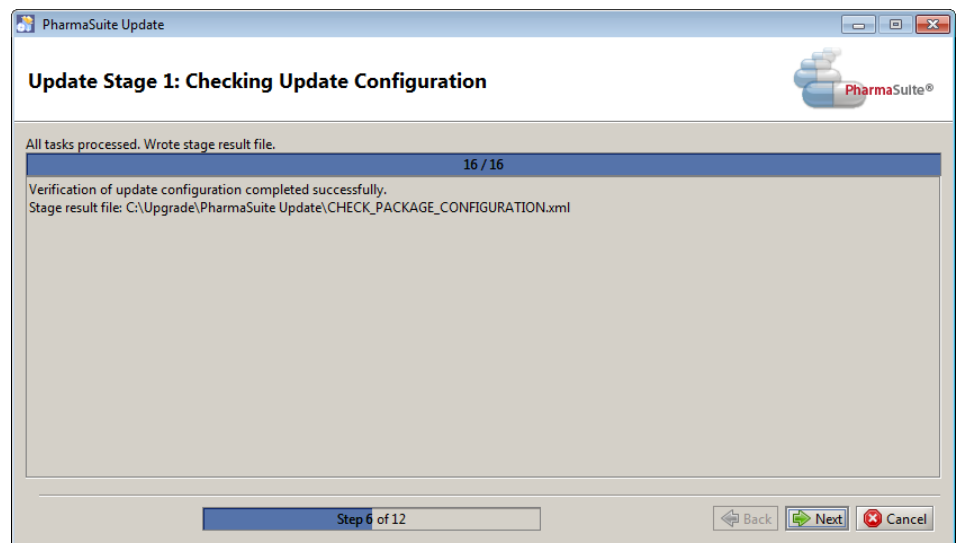


Figure 9: Checking Update Configuration

8. The system starts stage 2 of the update process and determines which changes need to be made to the various system objects. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the update installer and is called *COMPUTE\_PLANNED\_CHANGES.xml*. Click the **Next** button to continue.

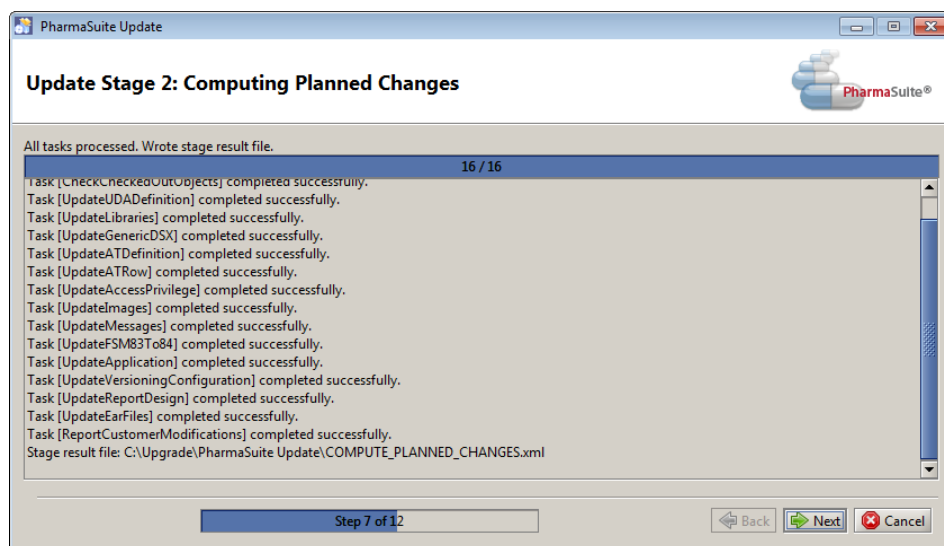


Figure 10: Computing Planned Changes

9. The system starts stage 3 of the update process and determines if executing the planned changes would lead to conflicts on the target system. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the update installer and is called *COMPARE\_WITH\_TARGET\_SYSTEM.xml*. Click the **Next** button to continue.

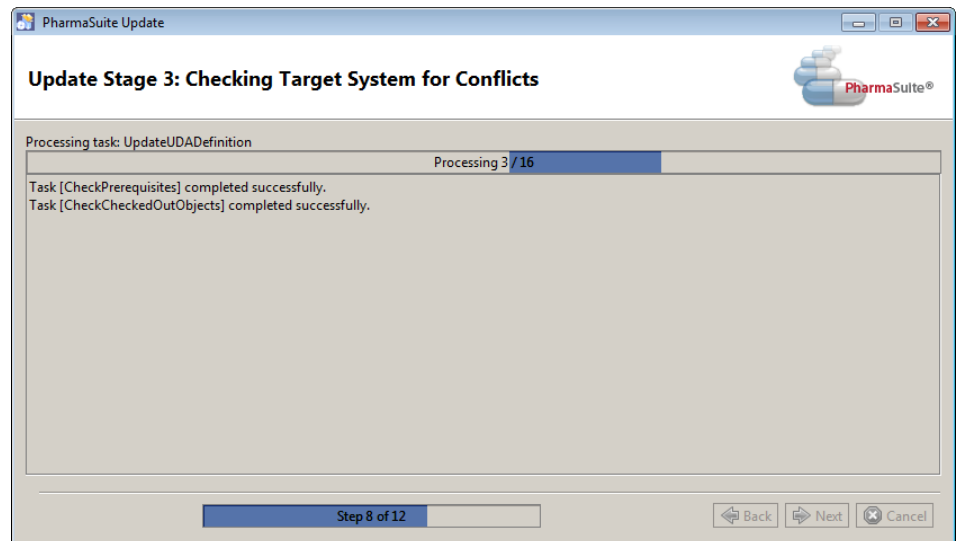


Figure 11: Checking Target System for Conflicts - in progress

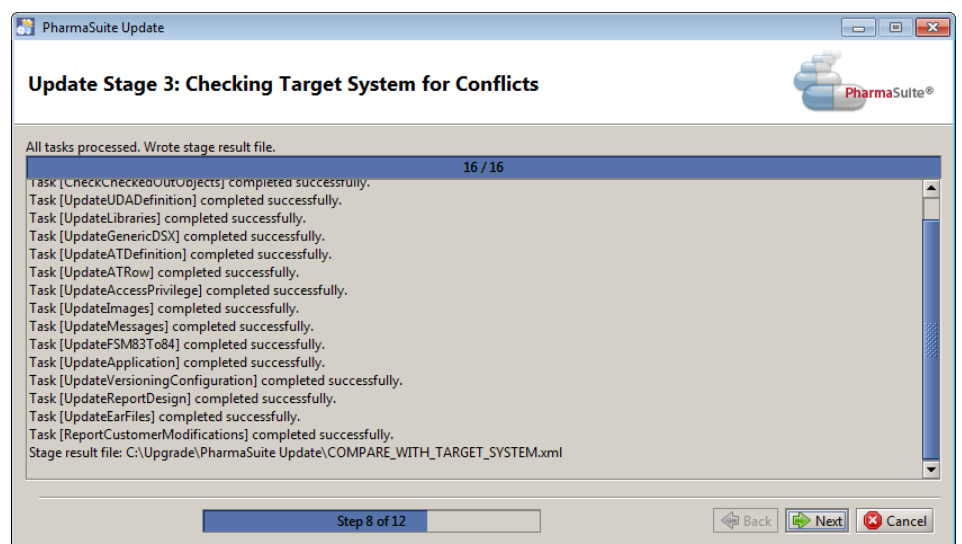


Figure 12: Checking Target System for Conflicts - completed

10. The system starts stage 4 of the update process and performs all changes that were identified in the previous stages. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the update installer and is called *UPDATE\_TARGET\_SYSTEM.xml*. Click the **Next** button to continue.

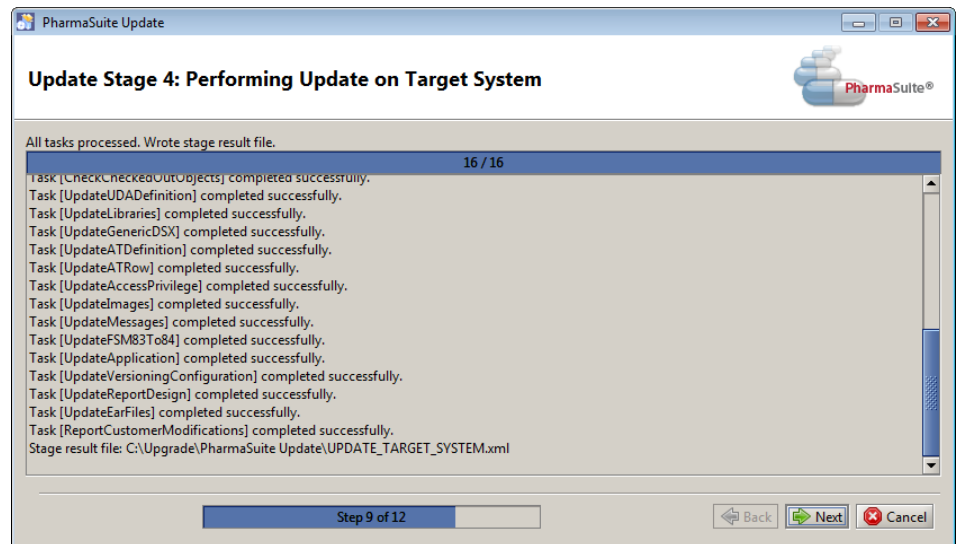


Figure 13: Performing Update on Target System



11. The system starts stage 5 of the update process to verify that the update was successful and all changes were applied. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the update installer and is called *VERIFY\_UPDATED\_SYSTEM.xml*. Click the **Next** button to continue.

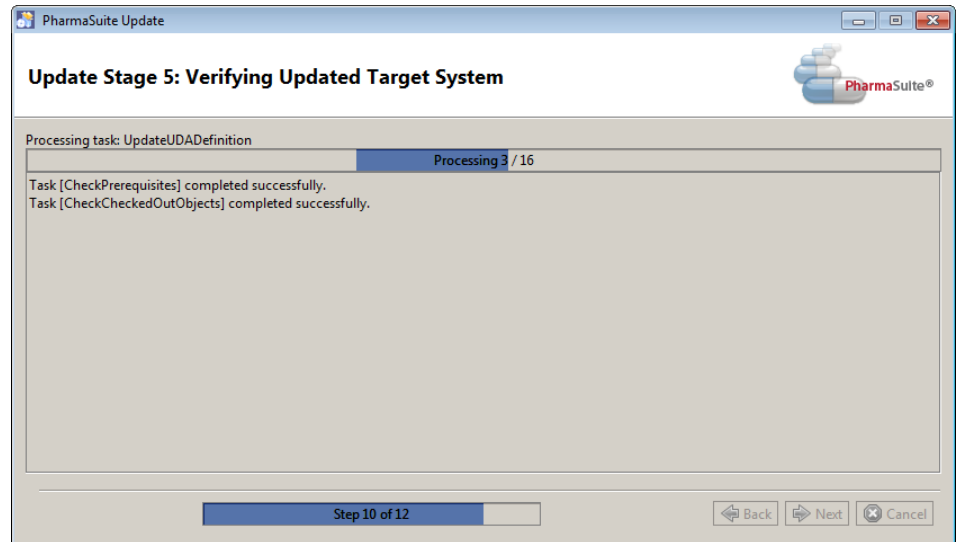


Figure 14: Verifying Updated Target System - in progress

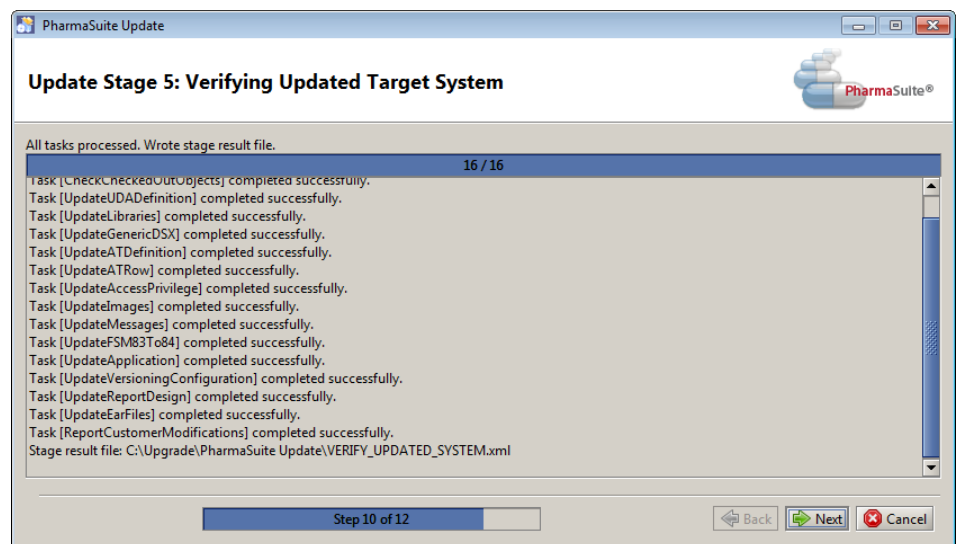


Figure 15: Verifying Updated Target System - completed

12. Click the **Finish** button to close the update wizard.

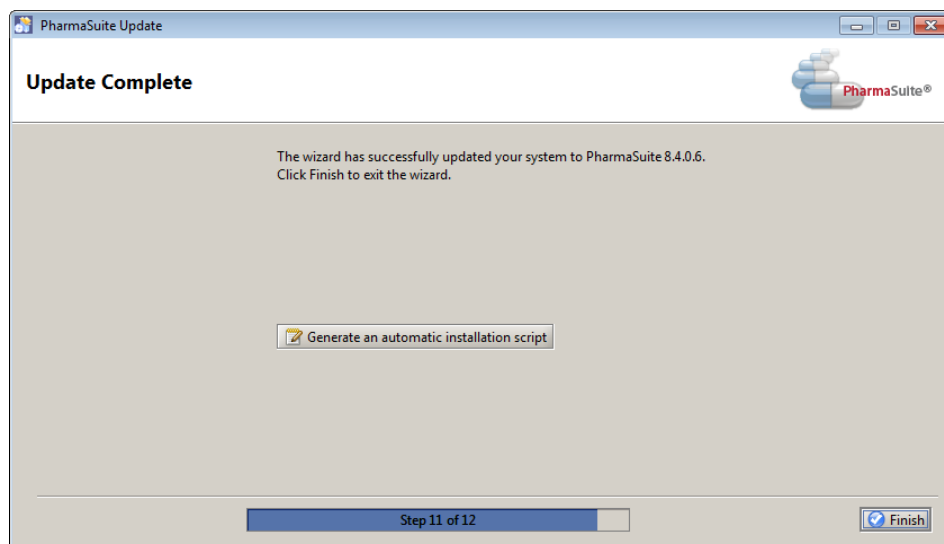


Figure 16: Update Complete

**TIP**

The **Generate an automatic installation script** button is only required for the preparation of an automated installation (page [43](#)).

13. Proceed with the second phase of the update process and migrate the data (page [25](#)).

## Migrating the Data

Please use the prerequisites and information checklists to verify that all preparatory steps have been completed and you have all information available you will need during the migration process.

The following two checklists cover all preparatory steps and information required for migrating the data of a PharmaSuite system

### INFORMATION CHECKLIST

Please prepare the information you will need during the data migration process:

	Information	Your Notes	Done?
1	Application server: Remote URL		
2	Application server: HTTP URL		
3	Application server: Login ID (user name) with at least <b>MinimalAccess</b> privilege		
4	Application server: Password		
5	System configuration considers the following settings: ActiveMQ and the PharmaSuite upgrade engine run with the 64-bit version of Java 1.8.0_144. Each PharmaSuite client (including Shop Operations Server) is only supported for the 32-bit version of Java 1.8.0_144. Make sure that the specified programs use a JAVA_HOME environment variable pointing to the correct Java version.		

### TIPS

Please do not put your password in writing, only include a hint to it in your notes.

To configure which default values are preset for settings such as the application server information, navigate to the downloaded *update-package-data-<version>.jar\resources* directory and open the *userInputSpec.xml* file in a text editor.

You can define a default by editing or defining the `set` attributes of the `spec txt` tags:

```
<spec txt="Remote URL" size="50" set="remote://MyServer:8080" />
```

The following values can be preset with a default:

- Remote URL
- Server HTTP URL
- Password
- Confirm password

### PREREQUISITES CHECKLIST

Before you start the data migration wizard, check the prerequisites:

	Prerequisite	Your Notes	Done?
1	User who will perform the data migration has Administrator rights to the operating system.		
2	User whose credentials will be used to access the application server during the data migration has at least the <b>MinimalAccess</b> privilege to PharmaSuite and references the <b>DefaultConfiguration</b> application.		
3	Update wizard has been run and the system artifact baseline has been updated successfully from PharmaSuite 8.3 to PharmaSuite 8.4.		
4	Data migration wizard ( <i>update-package-data-&lt;version&gt;.jar</i> ) downloaded from the Rockwell Automation Download Site (page 91) and saved locally or in a network directory. <version> stands for the final version and build number of PharmaSuite Data Migration.		

## Running the Data Migration Wizard

To execute the data migration, proceed as follows:

1. You need to run the data migration wizard with a startup parameter to make sure you have sufficient memory assigned to the process. In the directory where you have saved the *update-package-data-<version>.jar* file before, open a command prompt as administrator and type as follows:

```
java -Xmx1024m -jar update-package-data-<version>.jar
```

### TIP

If you wish to change the global block sizes used by all data migration tasks, run the data migration wizard with an additional VM argument as startup parameter. Example:

```
java -Xmx1024m -D loadObjectsBlockSize=100 -jar
update-package-data-<version>.jar
```

runs all data migration tasks that load objects in a blockwise fashion and loads only 100 objects at a time instead of 1000 objects. For details, see "Blockwise Update" (page 73).

Then press the ENTER key to start the update wizard.

### TIP

Please note that the command prompt needs to remain open during the entire run of the wizard. You can minimize it to the system tray but must not close it.

You can improve the performance of the wizard by running it as close to the application server as possible, ideally on the same machine as the application server, since the process involves a large number of middle-tier calls.

```
C:\Windows\System32\cmd.exe
2017-12-18 15:06:17,388 INFO StagePanel:255 - Task [UpdateFSM83To84] completed
successfully.
2017-12-18 15:06:17,388 INFO StagePanel:166 - Processing task [UpdateApplication]
2017-12-18 15:06:17,419 INFO StagePanel:255 - Task [UpdateApplication] completed
successfully.
2017-12-18 15:06:17,419 INFO StagePanel:166 - Processing task [UpdateVersioning
Configuration]
2017-12-18 15:06:17,435 INFO StagePanel:255 - Task [UpdateVersioningConfigurati
on] completed successfully.
2017-12-18 15:06:17,435 INFO StagePanel:166 - Processing task [UpdateReportDesi
gn]
2017-12-18 15:06:17,622 INFO StagePanel:255 - Task [UpdateReportDesign] complet
ed successfully.
2017-12-18 15:06:17,622 INFO StagePanel:166 - Processing task [UpdateEarFiles]
2017-12-18 15:06:18,543 INFO StagePanel:255 - Task [UpdateEarFiles] completed s
uccessfully.
2017-12-18 15:06:18,543 INFO StagePanel:166 - Processing task [ReportCustomerMo
difications]
2017-12-18 15:06:18,543 INFO StagePanel:255 - Task [ReportCustomerModifications
] completed successfully.
2017-12-18 15:06:18,558 INFO StagePanel:255 - Stage result file: C:\Upgrade\Pha
rmaSuite Update\VERIFY_UPDATED_SYSTEM.xml
c:\Upgrade>java -Xmx1024m -jar update-package-data-8.4.0.6.jar
```

Figure 17: Command prompt with startup parameter

2. Wait until the welcome dialog opens. Click the **Next** button to continue.

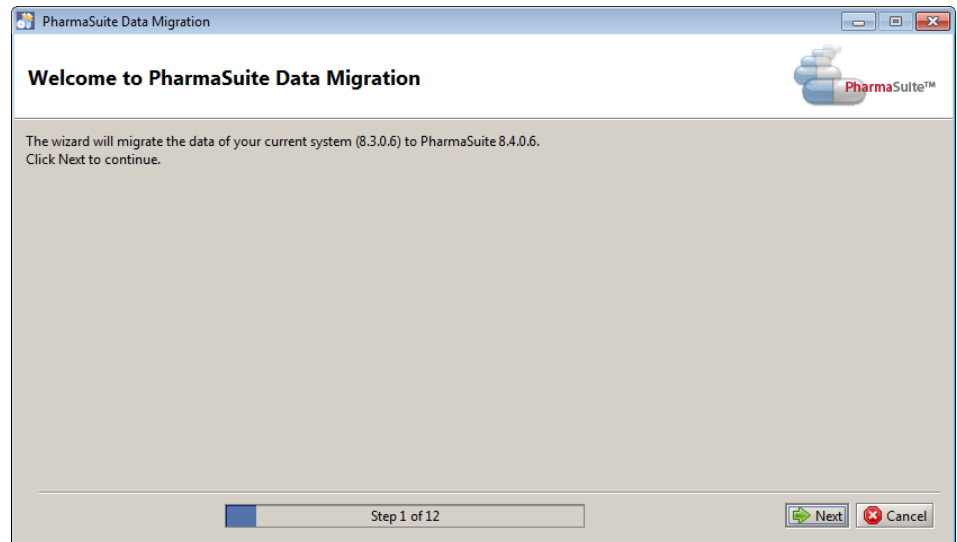


Figure 18: Welcome

3. Select the directory to which you wish to extract the data migration installer. To change the default path, click the **Browse** button and select another directory.

**TIP**

Make sure to extract the data migration wizard to another directory than the update wizard in order to avoid overwriting.

Click the **Next** button to continue.

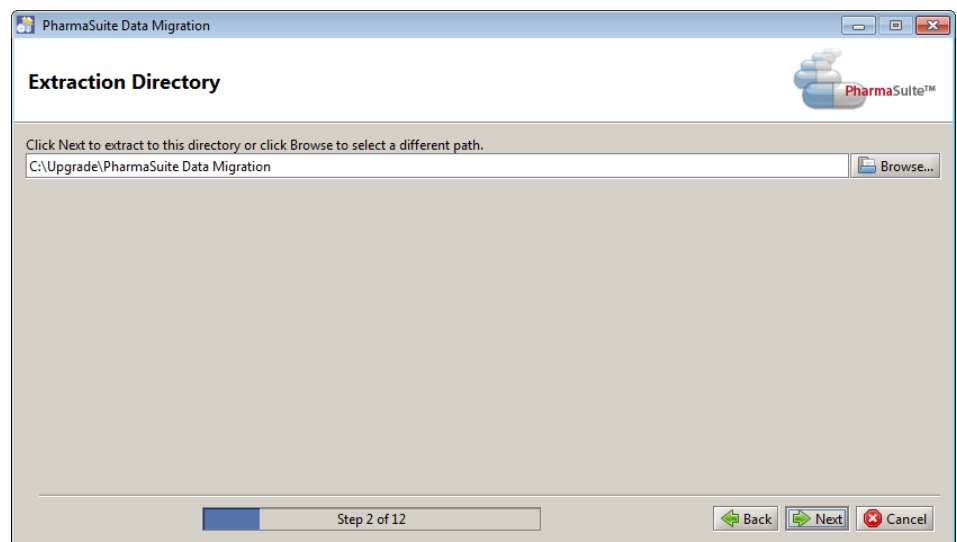


Figure 19: Extraction Directory

4. The system lists the packages that are available for extraction. The **Update Engine** package is required to run the data migration and thus cannot be unselected. You only need to extract the **SDK for the Update Package** if you plan to adapt the upgrade installer (page 77) itself and build a new version of the data migration wizard. Select the packages you wish to extract. Click the **Next** button to continue.

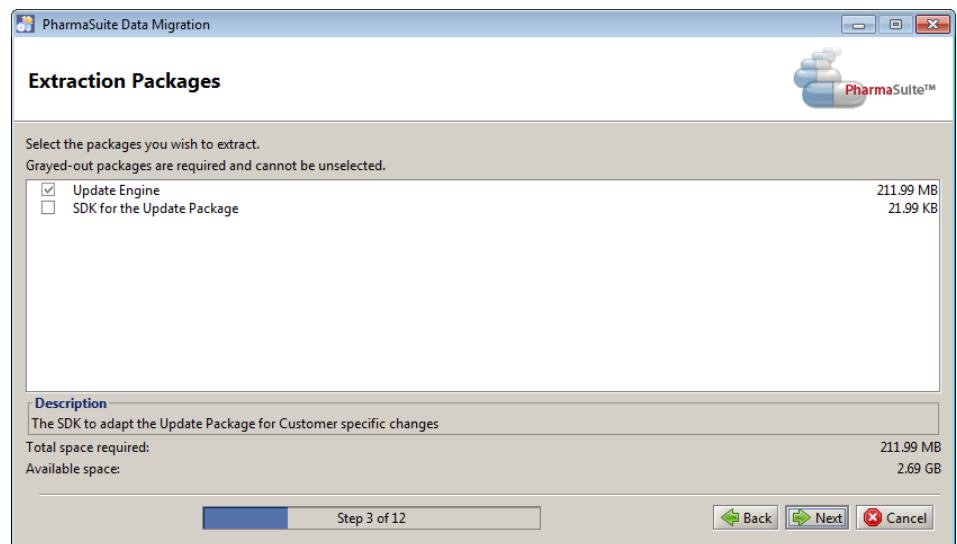


Figure 20: Extraction Packages

5. Wait until all packages have been extracted. Click the **Next** button to continue.

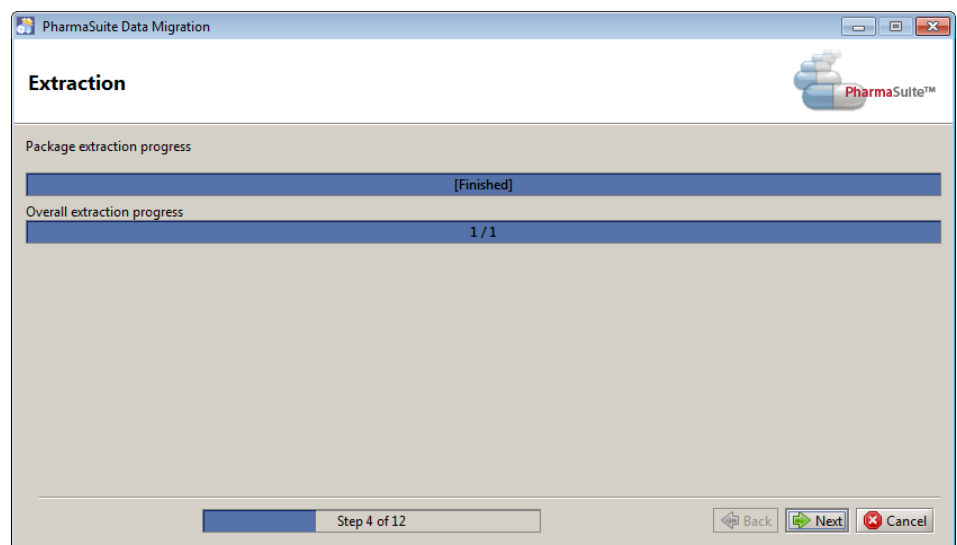


Figure 21: Extraction

6. Provide the data for the application server connection:

- **Remote URL** of your application server (here: **remote://ftpsmigration04:8080**).
- **Server HTTP URL** of your application server (here: **http://ftpsmigration04:8080**).
- **User** (login name) and **Password** of a user with suitable access privileges to perform the update (at least **MinimalAccess**).

**TIP**

Please note that the preset values in the input boxes can be edited. To change the presets themselves, you need to adjust the default values before you run the migration wizard. They are defined in the *userInputSpec.xml* file in the downloaded *update-package-data-<version>.jar\resources* directory.

Click the **Next** button to continue.

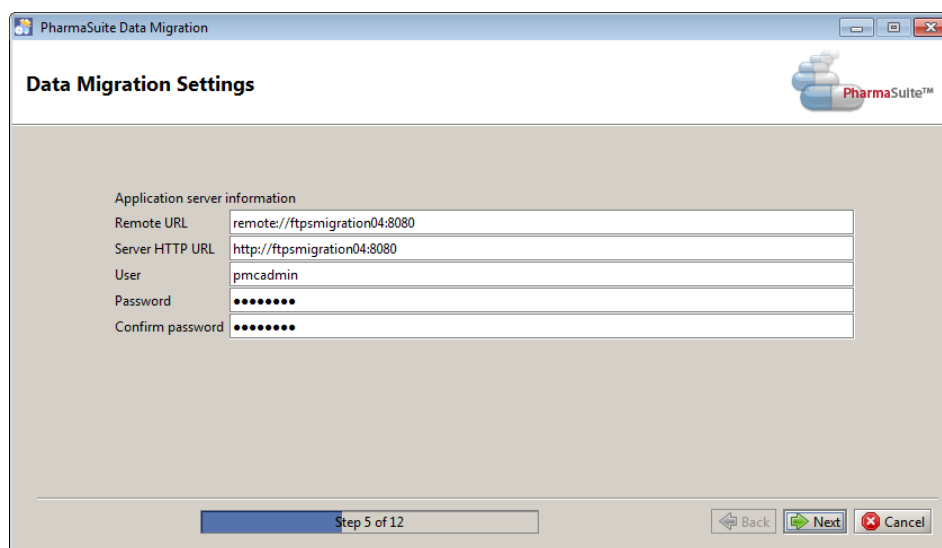


Figure 22: Data Migration Settings

**IMPORTANT**

If you have accidentally entered an incorrect URL in this step and clicked the **Next** button, the system informs you that it cannot connect to the application server. Please note that you cannot use the **Back** button to return to the step for correcting your input, but have to click the **Cancel** button to terminate the wizard and then restart it (with the startup parameters). However, if an erroneous input is located in the **User** or **Password** boxes, you can use the **Back** button to step back and correct your input.



7. The system starts stage 1 of the data migration process and checks the migration configuration.

**TIPS**

Please note that you cannot cancel a data migration stage in mid-process. When you click the **Cancel** button, the system first completes all tasks of the stage to ensure data and log file consistency before it terminates the data migration wizard.

Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the data migration installer and is called *CHECK\_PACKAGE\_CONFIGURATION.xml*. Click the **Next** button to continue.

**TIP**

Look at the individual result files (page 38) of the 5 stages to see if the respective tasks have identified issues with the objects they handle.

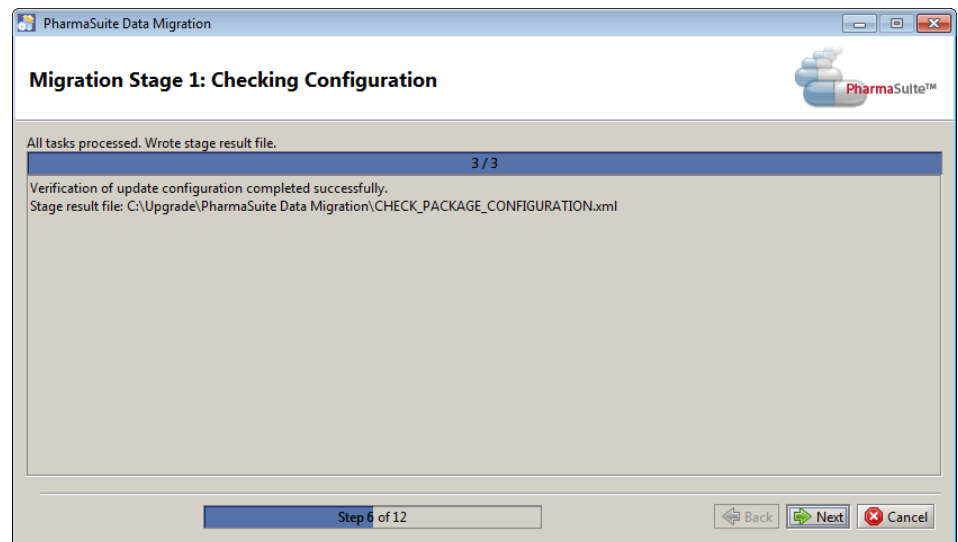


Figure 23: Checking Configuration

8. The system starts stage 2 of the data migration process and determines which changes need to be made to the various system objects. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the data migration installer and is called *COMPUTE\_PLANNED\_CHANGES.xml*. Click the **Next** button to continue.

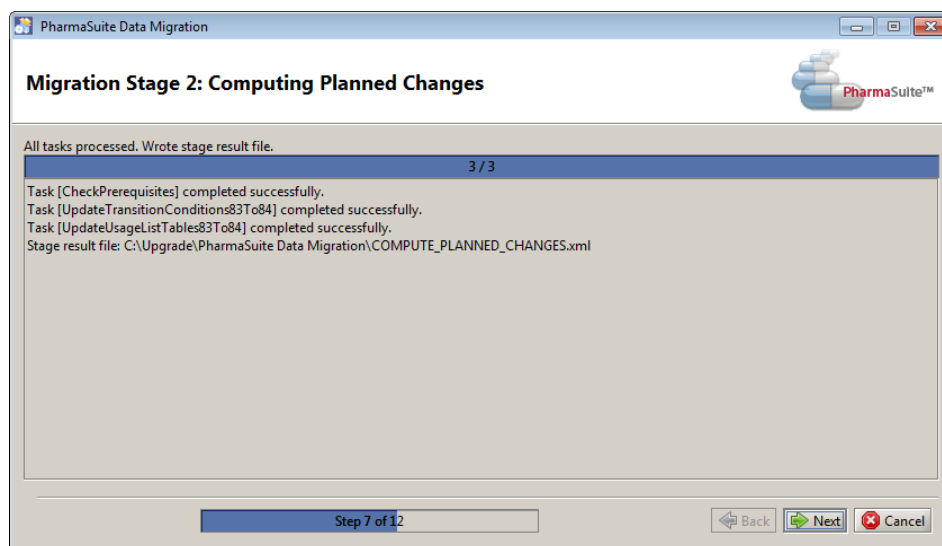


Figure 24: Computing Planned Changes

9. The system starts stage 3 of the data migration process and determines if executing the planned changes would lead to conflicts on the target system. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the data migration installer and is called *COMPARE\_WITH\_TARGET\_SYSTEM.xml*. Click the **Next** button to continue.

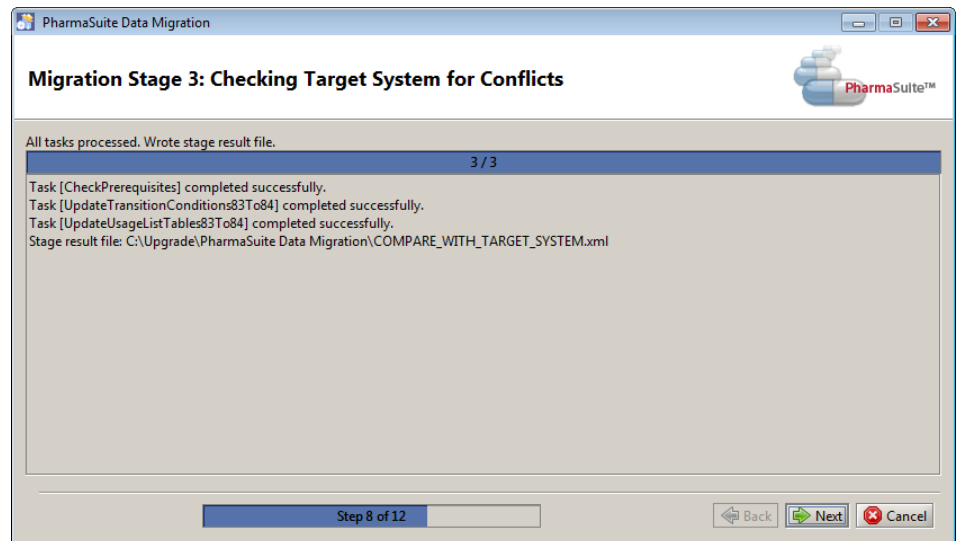


Figure 25: Checking Target System for Conflicts

10. The system starts stage 4 of the data migration process and performs all changes that were identified in the previous stages. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the data migration installer and is called *UPDATE\_TARGET\_SYSTEM.xml*. Click the **Next** button to continue.

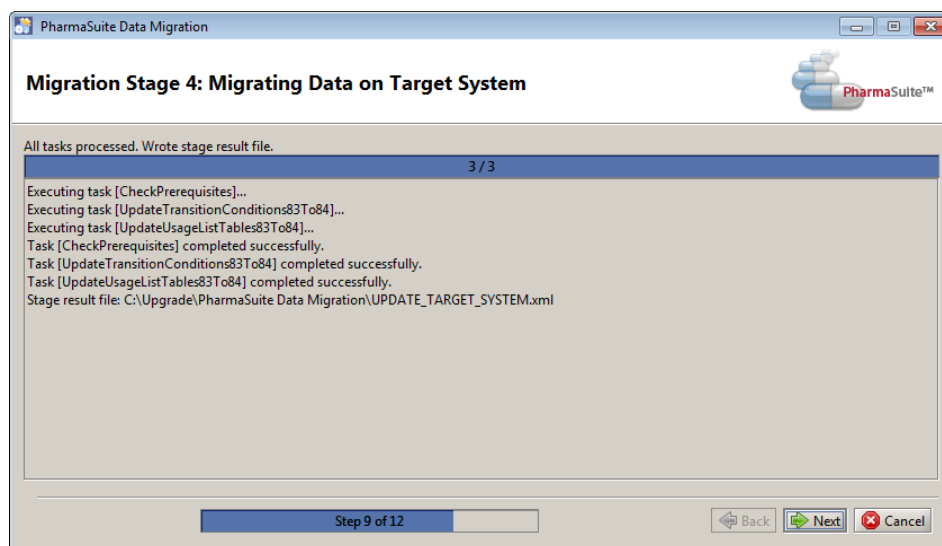


Figure 26: Migrating Data on Target System

11. The system starts stage 5 of the data migration process to verify that the migration was successful and all changes were applied. Wait until all tasks have completed processing. Then the system indicates the successful completion of the stage in the list box, along with the path to the stage result file. It is located in the directory to which you have extracted the data migration installer and is called *VERIFY\_UPDATED\_SYSTEM.xml*. Click the **Next** button to continue.

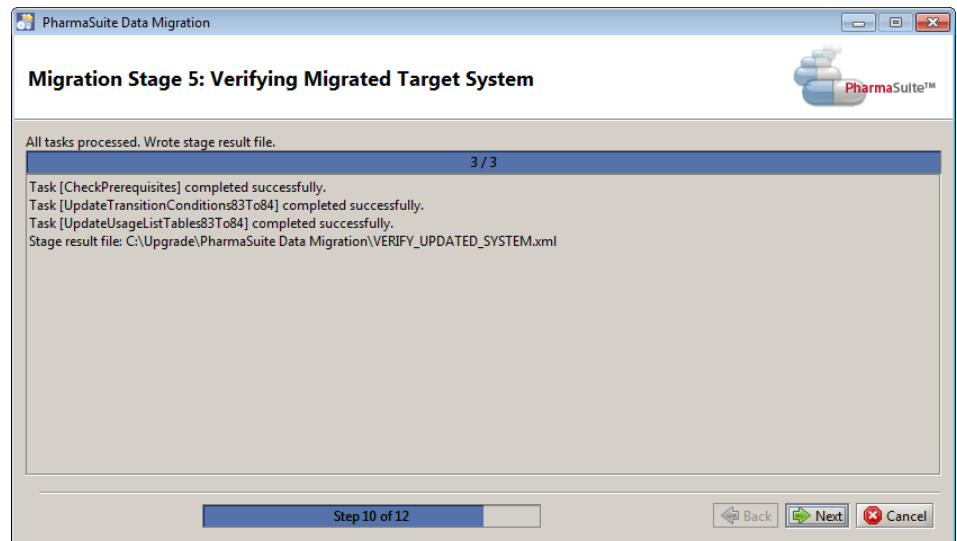


Figure 27: Verifying Migrated Target System

12. Click the **Finish** button to close the data migration wizard.

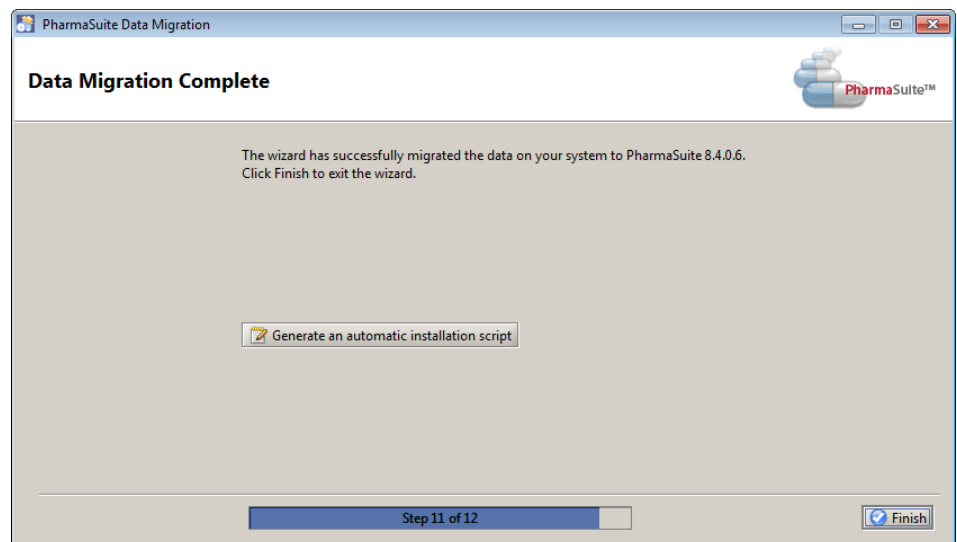


Figure 28: Data Migration Complete

**TIP**

The **Generate an automatic installation script** button is only required for the preparation of an automated installation (page 43).

13. Check the ActiveMQ configuration for PharmaSuite in *c:\apache-activemq-5.15.0\conf\activemq.xml*:  
Check the dead letter strategy to discard expired messages.  
These messages are no longer needed by PharmaSuite and would lead to out-of-memory situations if not discarded.  
Make sure that the **<policyEntries>** section contains only one **<policyEntry queue="">** tag with the following entry:  

```
<!-- Tell the dead letter strategy not to process expired
      messages so that they will just be discarded instead
      of being sent to the DLQ. -->
<deadLetterStrategy>
  <sharedDeadLetterStrategy processExpired="false" />
</deadLetterStrategy>
```
14. If the upgrade of your PharmaSuite system also includes the upgrade of phase building blocks, make sure you have updated all affected artifacts of the phase building blocks. This is especially relevant when you install MR+ revisions (page 36). Then proceed with removing duplicate classes from the classpath (page 37). Otherwise, restart the services of all PharmaSuite-related event sheets, which you stopped before running the update wizard.  
For more information on event sheets see "Technical Manual Administration" [A4] (page 91).

## Installing MR+ Revisions of Building Blocks

### TIP

An MR+ revision of a building block is an Extended Maintenance Release of a building block. For a detailed definition, see "PharmaSuite Building Blocks - Compatibility Matrix" [A10] (page 91).

MR+ revisions may require an update of phase-specific DSX files (e.g. report designs, message packs).

For more information, please refer to "Installing MR+ Revisions of Building Blocks" in "Technical Manual Installation - Building Block" [A9] (page 91).

## Removing Duplicate Classes from the Classpath

If the upgrade of your PharmaSuite system also includes the upgrade of phase building blocks, you need to perform a manual migration task on the upgraded system to remove duplicate classes from the classpath.

### TIP

After you have removed the duplicate classes, make sure to restart your Shop Operations Servers (SOS services) in order to load the correct JAR files into the classpath.

- If the 1<sup>st</sup> or 2<sup>nd</sup> digit of the version number of a JAR file available in Process Designer is increased, the libraries can exist in parallel (e.g. eqm-phase-shared-ai-1.0.0.9.jar and eqm-phase-shared-ai-1.1.0.9.jar).  
**This does not apply to files of the DCS Adapter.** The old Library object must be deleted in any case.
- If the 3<sup>rd</sup> or 4<sup>th</sup> digit of the version number of a JAR file available in Process Designer is increased (e.g. eqm-phase-shared-ai-1.0.0.9.jar and eqm-phase-shared-ai-1.0.1.5.jar), you must delete the old Library object.

Which of the installed libraries are affected, depends on your installation. To retrieve the affected libraries, use the phase manager tool.

1. In Process Designer, run the **mes\_PhaseLibManager** form to start the phase manager.
2. Navigate to the **Manage Basic Phases** tab.
3. Click the **Info (installed phases)** button to display detailed information about all installed basic phases and libraries. The system displays the following information:
  - In case there are no duplicate libraries installed:  
**No issues with duplicate libraries found.**
  - In case there are duplicate libraries installed:  
**Potential duplicate library issues found for:**  
**<Library name> with version: [<old version>], [<new version>]**
  - The output always provides a list of all installed phase building blocks and libraries including their versions.

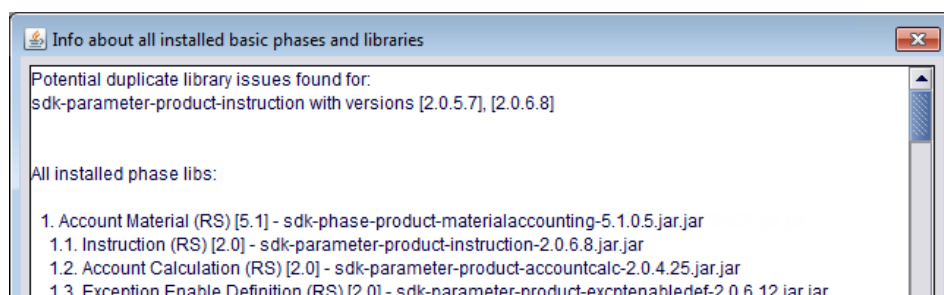


Figure 29: Example of a duplicate library

- If you wish to retain the information listed in the output, click the **Copy to clipboard** button to copy the information and then paste it into a system-external text editor.
4. Close the phase manager, then close the form.

## Reporting Upgrade Results

While the update and migration wizards run, the system writes dedicated stage reports in XML format at the end of each stage, which document the results of the update or migration processes. The reports contain information on all the tasks performed within a stage. They are located in the directory to which you have extracted the update or migration wizard.

### TIP

Please note that the wizards generate reports that have the same names but differ in content. For this reason, during the extraction steps of the wizard runs, the system suggests two different, unambiguously named extraction locations (*..\PharmaSuite Update* and *..\PharmaSuite Data Migration*).

The stage result reports are named as follows:

- **Stage 1** (Checking Update Configuration)/(Checking Configuration): *CHECK\_PACKAGE\_CONFIGURATION.xml*
- **Stage 2** (Computing Planned Changes): *COMPUTE\_PLANNED\_CHANGES.xml*
- **Stage 3** (Checking Target System for Conflicts): *COMPARE\_WITH\_TARGET\_SYSTEM.xml*
- **Stage 4** (Performing Update on Target System)/(Migrating Data on Target System): *UPDATE\_TARGET\_SYSTEM.xml*
- **Stage 5** (Verifying Updated Target System)/(Verifying Migrated Target System): *VERIFY\_UPDATED\_SYSTEM.xml*



The following XML elements are used:

- **StageResult**: General information about the result of a stage.
  - Attributes
    - **baseVersion**: PharmaSuite version from which the upgrade installer upgrades.
    - **endTime**: Completion time.
    - **hasErrors**: Overall result (false, true).
    - **javaVersion**: Vendor and version of the JVM on which the upgrade installer runs.
    - **localhost**: Name of the machine on which the upgrade installer runs.
    - **localUser**: User account under which the upgrade installer runs.
    - **newVersion**: PharmaSuite version to which the upgrade installer upgrades.
    - **operatingSystem**: Name and version of the operating system on which the upgrade installer runs.
    - **stage**: Name of the stage.
    - **startTime**: Stage start time (e.g. 2012-05-21T09:51:47.199+02:00).
    - **targetProductionCentreVersion**: FactoryTalk ProductionCentre version installed on the system to be upgraded.
    - **updateServer**: URL of the server to be upgraded.
    - **updateUser**: FactoryTalk ProductionCentre user used by the upgrade installer to perform the upgrade.
    - **xmlns**: XML schema definition and name space of this XML file.
  - Nested elements
    - **TaskResult**
- **TaskResult**: Basic information about the result of an upgrade task.
  - Attributes
    - **endTime**: Completion time.
    - **hasErrors**: Overall result (false, true).
    - **startTime**: Start time (e.g. 2012-01-18T15:16:34.805+01:00).
    - **taskClassname**: Internal task name.
  - Nested elements
    - **TaskResultItem**

- **TaskResultItem:** Basic information about the result of updating an item within a task.
  - **Attributes**
    - **affectedObject:** Object name (e.g. pmc\_message, Check PharmaSuite version).
    - **changeLevel:** System, ObjectType, Object, Attribute, Data, or Other.
    - **changeType:** Add, Delete, Overwrite, Merge, DataTransformation, or Other.
  - **Nested elements**
    - **TaskResultItem**
      - **Error:** Message describing an error condition (e.g. "Cannot update object [123] on the target system. [123] is marked for deletion, but its content was modified on the target system.").
      - **Warning:** Message describing a warning condition (e.g. "UDA definition [123] is not needed any longer. However, since data is generally not deleted, the UDA definition remains unchanged.").
      - **Info:** Specific information about the item (e.g. Expected PharmaSuite version is PharmaSuite x.x.x.x).

Example report:

```
<?xml version="1.0" encoding="UTF-8"?>
<StageResult startTime="2015-12-16T10:06:25.196+01:00" hasErrors="false"
  stage="COMPUTE_PLANNED_CHANGES"
  operatingSystem="Windows 7 v6.1 x86 "
  javaVersion="Oracle Corporation 1.7.0_60"
  localUser="Migration"
  localhost="MIGRATION04"
  baseVersion="PharmaSuite x.x.x.x"
  newVersion="PharmaSuite y.y.y.y"
  targetProductionCentreVersion="FactoryTalk ProductionCentre z.z.zzzzzz"
  updateUser="pmcadmin" updateServer="http://ftpsmigration04:8080"
  endTime="2015-12-16T10:06:25.232+01:00"
  xmlns="http://rockwell.com/mes/update/databeans">
  <TaskResult startTime="2015-12-16T10:06:25.201+01:00"
    endTime="2015-12-16T10:06:25.210+01:00"
    taskClassname="com.rockwell.mes.update.tasks.general.CheckPrerequisites"
    hasErrors="false">
    <TaskResultItem affectedObject="Check PharmaSuite version"
      changeLevel="System" changeType="Other">
      <Info>Expected PharmaSuite version is PharmaSuite x.x.x.x</Info>
    </TaskResultItem>
    <TaskResultItem affectedObject="Check ProductionCentre version"
      changeLevel="System" changeType="Other">
      <Info>Expected ProductionCentre version is
        FactoryTalk ProductionCentre z.z.zzzzzz</Info>
    </TaskResultItem>
  </TaskResult>
  <TaskResult startTime="2015-12-16T10:06:25.213+01:00"
    endTime="2015-12-16T10:06:25.218+01:00"
```

```

        taskClassname="com.rockwell.mes.update.tasks.onetime.
        EnableStatusOfS88EqmClasses71To80" hasErrors="false">
    <TaskResultItem affectedObject="Equipment Classes" changeLevel="Object"
        changeType="DataTransformation">
        <Info>All equipment classes will be migrated to enable status
        management.</Info>
    </TaskResultItem>
</TaskResult>
<TaskResult startTime="2015-12-16T10:06:25.219+01:00"
    endTime="2015-12-16T10:06:25.219+01:00"
    taskClassname="com.rockwell.mes.update.tasks.onetime.
    EnableStatusOfS88EqmEntities71To80" hasErrors="false">
    <TaskResultItem affectedObject="Equipment Entities" changeLevel="Object"
        changeType="DataTransformation">
        <Info>All equipment entities will be migrated to enable status
        management.</Info>
    </TaskResultItem>
</TaskResult>
<TaskResult startTime="2015-12-16T10:06:25.220+01:00"
    endTime="2015-12-16T10:06:25.221+01:00"
    taskClassname="com.rockwell.mes.update.tasks.onetime.
    UpdateTransitionLocksOfMasterRecipes71To80" hasErrors="false">
    <TaskResultItem affectedObject="Transitions of Master Recipes and Master
    Workflows" changeLevel="Object" changeType="DataTransformation">
        <Info>All transitions of recipe and workflow structures will be migrated to
        set the lock flag.</Info>
    </TaskResultItem>
</TaskResult>
<TaskResult startTime="2015-12-16T10:06:25.222+01:00"
    endTime="2015-12-16T10:06:25.222+01:00"
    taskClassname="com.rockwell.mes.update.tasks.onetime.
    UpdateTransitionLocksOfBuildingBlocks71To80" hasErrors="false">
    <TaskResultItem affectedObject="Transitions of Building Blocks"
        changeLevel="Object" changeType="DataTransformation">
        <Info>All transitions of building blocks will be migrated to set the
        lock flag.</Info>
    </TaskResultItem>
</TaskResult>
<TaskResult startTime="2015-12-16T10:06:25.224+01:00"
    endTime="2015-12-16T10:06:25.224+01:00"
    taskClassname="com.rockwell.mes.update.tasks.onetime.
    UpdateExportedStatusOfOrders71To80" hasErrors="false">
    <TaskResultItem affectedObject="Batch Orders, Device Orders, and
    Workflows" changeLevel="Object" changeType="DataTransformation">
        <Info>All orders and workflows will be migrated to set their Exported
        for archive status to Not exported.</Info>
    </TaskResultItem>
</TaskResult>
<TaskResult startTime="2015-12-16T10:06:25.227+01:00"
    endTime="2015-12-16T10:06:25.228+01:00"
    taskClassname="com.rockwell.mes.update.tasks.onetime.
    UpdateProductionRelevantFlagOfWorkflowOrders71To80" hasErrors="false">
    <TaskResultItem affectedObject="Workflows" changeLevel="Object"
        changeType="DataTransformation">
        <Info>All workflows will be migrated to update their Production-relevant
        settings.</Info>
    </TaskResultItem>
</TaskResult>
<TaskResult startTime="2015-12-16T10:06:25.229+01:00"
    endTime="2015-12-16T10:06:25.230+01:00"
    taskClassname="com.rockwell.mes.update.tasks.onetime.
    UpdateSoftReferencesForAppendedWFs71To80" hasErrors="false">

```

```
<TaskResultItem affectedObject="Appended Workflow Entries"
  changeLevel="Object" changeType="DataTransformation">
  <Info>The information about appended workflows will be migrated to contain
    soft references to the corresponding order step and other relevant
    objects.</Info>
</TaskResultItem>
</TaskResult>
<TaskResult startTime="2015-12-16T10:06:25.231+01:00"
  endTime="2015-12-16T10:06:25.232+01:00"
  taskClassname="com.rockwell.mes.update.tasks.onetime.
  UpdateSoftReferencesForEqmLogbook71To80" hasErrors="false">
  <TaskResultItem affectedObject="Equipment Logbook Entries"
    changeLevel="Object" changeType="DataTransformation">
    <Info>All equipment logbook entries will be migrated to contain soft
      references to the context-relevant objects.</Info>
  </TaskResultItem>
</TaskResult>
</StageResult>
```

## Logging the Upgrade

When a wizard runs, it writes a log file that lists the actions performed during its individual stages. The log files of the wizards are called *update.log* and are located in the directories to which you have extracted the wizards. You can also access and adjust the logging configuration of the upgrade installer individually for the update or the data migration after you have run the **Extraction** step of the respective wizard. To log more details during the five stages of the update or data migration processes, proceed as follows:

1. In the directory to which you have extracted the update or data migration packages, open the *log4jupdater.properties* file in a text editor.
2. In the *Log more details for the FTPS update* section prefix the `log4j.logger.com.rockwell.mes.update = INFO` line with a hash sign (#) to comment it out and remove the leading hash sign from the following line (`log4j.logger.com.rockwell.mes.update = DEBUG`) to uncomment it instead.
3. Save and close the file and proceed with running the wizard to which it applies.

### TIP

Please note that you have to apply and save the change of the log level before you start running the five upgrade stages in your wizard, otherwise the change will not take effect.

## Automated Installation of the Upgrade

The PharmaSuite upgrade installer supports the automated (unattended) installation technique as automated wizard installation and as console mode installation. For technical details of the underlying installer tool, see "Unattended Installations (IzPack)" [D1] (page 92).

**TIP**

The automated installation is only intended to be used for automated testing purposes.

### Automated Wizard Installation

An automated wizard installation allows to execute an unattended installation based on a previously performed system update (page 15) or data migration (page 27) using the corresponding wizard.

To create a script for an automated update wizard installation, proceed as follows:

1. Run the update wizard with your settings required for an automated installation.
2. In the final step (**Update Complete**), click the button to generate an automated installation script.
3. Save the file as *AutomatedInstallation.xml*.
4. Click the **Finish** button to close the update wizard.

To create a script for an automated data migration, perform the analogous steps with the data migration wizard.

**TIP**

Make sure to save the data migration-related automated installation script to another directory than the update-related script in order to avoid overwriting.

To run an automated update wizard, proceed as follows:

1. You need to run the automated update wizard with a startup parameter to make sure you have sufficient memory assigned to the process. In the directory where you have saved the *update-package-builder-[<version>.jar](#)* file before, open a command prompt as administrator and type as follows:

```
java -Xmx1024m -jar update-package-builder-<version>.jar AutomatedInstallation.xml
```

**TIP**

For the data migration, you can also configure the block sizes used by the data migration tasks with additional VM arguments as startup parameters:

-DloadObjectsBlockSize and/or -DsqlUpdateBlockSize. For details, see "Blockwise Update" (page [73](#)).

Then press the ENTER key to start the automated update wizard.

**TIP**

Please note that the command prompt needs to remain open during the entire run of the automated update wizard.

2. The automated update wizard indicates its completion with the following message:

```
Stage [VERIFY_UPDATED_SYSTEM] completed successfully.  
[Automated installation done ]
```

or

```
...  
[Automated installation FAILED! ]
```

To run an automated data migration wizard, perform the analogous steps for the *update-package-data-[<version>.jar](#)* file.

## Console Mode Installation

A console mode installation allows to execute an unattended installation based on a properties file.

To create and configure a template properties file for an automated console mode update installation, proceed as follows:

1. In the directory where you have saved the *update-package-builder-[<version>.jar](#)* file before, open a command prompt as administrator and type as follows:

```
java -Xmx1024m -jar update-package-builder-<version>.jar  
-options-template ConsoleModeInstallation.properties
```

### TIP

For the data migration, you can also configure the block sizes used by the data migration tasks with additional VM arguments as startup parameters:

`-DloadObjectsBlockSize` and/or `-DsqlUpdateBlockSize`. For details, see "Blockwise Update" (page [73](#)).

2. Configure the settings of the *ConsoleModeInstallation.properties* file with a text editor. Use the values you usually provide in the **Extraction Directory** and **Update Settings** steps of the update wizard (page [15](#)).

### TIP

The properties file cannot process backslashes. Make sure to replace backslashes used in file paths with forward slashes.

To create and configure a template properties file for an automated console mode data migration, perform the analogous steps for data migration (page [27](#)).

To run an automated console mode update, proceed as follows:

1. You need to run the console mode update with a startup parameter to make sure you have sufficient memory assigned to the process. In the directory where you have saved the *update-package-builder-[<version>.jar](#)* file before, open a command prompt as administrator and type as follows:

```
java -Xmx1024m -jar update-package-builder-<version>.jar  
-options ConsoleModeInstallation.properties
```

**TIP**

For the data migration, you can also configure the block sizes used by the data migration tasks with additional VM arguments as startup parameters:

`-DloadObjectsBlockSize` and/or `-DsqlUpdateBlockSize`. For details, see "Blockwise Update" (page [73](#)).

Then press the ENTER key to start the automated console mode update.

**TIP**

Please note that the command prompt needs to remain open during the entire run of the automated console mode update.

2. The automated console mode update indicates its completion with the following message:

```
Stage [VERIFY_UPDATED_SYSTEM] completed successfully.  
[Console installation done ]
```

or

```
...  
[Console installation FAILED! ]
```

To run an automated console mode data migration, perform the analogous steps for the *update-package-data-[<version>.jar](#)* file.



## Technical Details of a System Update

This section provides technical details of a system update. The update affects DSX files, JAR files linked with DSX files (e.g. libraries), and other files (e.g. EAR files).

A general update strategy (page 47) applies to all of these artifacts, which are processed by specific update tasks (page 50).

### TIPS

DSX objects from the **NEW\_VERSION** folder need to be imported as they are, since any changes applied to them before the import are ignored during the import process. If you need to apply changes to such an object, use the API of FactoryTalk ProductionCentre to manipulate the object in the database after it has been imported.

In **Stage 3** (Checking Target System for Conflicts) and **Stage 5** (Verifying Updated Target System), all exportable ATRow objects from the target system are exported, no matter if they will be migrated or not. This includes ATRow objects of customer-specific ATDefinitions with the **EXPORTABLE** data management type. If there are such ATDefinitions with many ATRows, this could affect the migration performance. To avoid this, consider to change the data management type of such ATDefinitions to **STATIC** before the system update and change it back to **EXPORTABLE** afterwards. Please do **not** do this for system-specific ATDefinitions as this may affect migration.

### General Update Strategy

Most objects can be updated according to the general update strategy outlined in the subsequent table. It lists the cases of differences that may occur during the comparison of the **BASE** standard version of PharmaSuite with the **NEW** standard version and the configured and possibly extended **TARGET** version of a customer.

Case	BASE	NEW	TARGET	Result of Stage 2	Result of Stage 3
01	A	A	A	None	None
02	A	---	---	Delete	None
03	A	---	A	Delete	Delete
04	A	---	C	Delete	Conflict (page 49)
05	---	B	B	Add	None
06	---	B	---	Add	Add
07	---	B	C	Add	Conflict (page 49)
08	A	B	B	Overwrite	None

Case	BASE	NEW	TARGET	Result of Stage 2	Result of Stage 3
09	A	B	A	Overwrite	Overwrite
10	A	B	C	Overwrite	Conflict (page 49)
11	A	B	---	Overwrite	Conflict (page 49)
12	---	---	C	None	None, see Reporting Customer Modifications (page 71)
13	A	A	---	None	None, see Reporting Customer Modifications (page 71)
14	A	A	C	None	None

### LEGEND

The capital letters in the **BASE**, **NEW**, and **TARGET** columns represent different variants of the same object residing in either the **BASE**, **NEW**, or **TARGET** systems. "---" indicates that there is no variant of the object in the respective system. The differences between the object variants can come with new or changed functions or structures in the new PharmaSuite system or result from customer-specific extensions made to the target system.

**Result of Stage 2** (Computing Planned Changes) lists the actions to be potentially executed.

- **Add:** object will be added to the target system.
- **Delete:** object will be deleted from the target system.
- **None:** no change of the target system required.
- **Overwrite:** object will be replaced in the target system.

**Stage 3** (Checking Target System for Conflicts) lists the actions to be executed in **Stage 4** (Performing Update on Target System).

- **Add:** object will be added to the target system.
- **Conflict:** target system cannot be updated automatically. For details how to resolve the conflict, see the description of the object-specific upgrade task.
- **Delete:** object will be deleted from the target system.
- **None:** no change of the target system required.

For further details and specifics, see the descriptions of the object-specific upgrade tasks. Usually, conflicts will be resolved by manual merges and/or adaptations to the upgrade installer. The target system is not touched for resolving conflicts. For general guidance, see "How to Resolve Conflicts" (page 49).

## How to Resolve Conflicts

To resolve a conflict, you have to decide which version is required by your system after the update (**FINAL** column). Depending on the decision, either the **BASE** version or the **NEW** version needs to be modified accordingly (see "Adapting the Upgrade Installer" (page 77)).

The approaches listed below will fit for most objects. If an object requires a different solution, see the description of the object-specific upgrade task (e.g. FSM (page 62)).

- Resolve case 4 (BASE = A, NEW = ---, TARGET = C):

BASE	NEW	TARGET	FINAL	Modify BASE to	Modify NEW to
A	---	C	---	C	--- (= no change)
A	---	C	C	A (= no change)	C

- Resolve case 7 (BASE = ---, NEW = B, TARGET = C):

BASE	NEW	TARGET	FINAL	Modify BASE to	Modify NEW to
---	B	C	B	C	B (= no change)
---	B	C	C	--- (= no change)	C
---	B	C	Merge of B and C	C	Merge of B and C

- Resolve case 10 (BASE = A, NEW = B, TARGET = C):

BASE	NEW	TARGET	FINAL	Modify BASE to	Modify NEW to
A	B	C	B	C	B (= no change)
A	B	C	C	A (= no change)	C
A	B	C	Merge of B and C	C	Merge of B and C

- Resolve case 11 (BASE = A, NEW = B, TARGET = ---):

BASE	NEW	TARGET	FINAL	Modify BASE to	Modify NEW to
A	B	---	B	---	B (= no change)
A	B	---	---	A (= no change)	---

## Update Tasks

The update process performs the following update tasks. Their sequence is defined in the update-specific configuration file (*updateConfiguration.xml*).

1. Check prerequisites.  
Checks the general prerequisites for the update process.  
Checks if the installed versions of PharmaSuite and FactoryTalk ProductionCentre match the expected versions.
2. Checking for checked-out objects (page [51](#)).
3. Updating UDA definitions (page [51](#)).
4. Updating Library objects (page [53](#)).
5. Updating DSX objects (generic DSX) (page [53](#)).
6. Updating AT definitions (page [57](#)).
7. Updating ATRow objects (page [59](#)).
8. Updating Access privilege objects (page [61](#)).
9. Updating Image objects (page [61](#)).
10. Updating FSM objects (page [62](#)).
11. Updating Message objects (page [67](#)).
12. Updating Application objects (page [68](#)).
13. Updating Versioning configuration objects (page [68](#)).
14. Updating Report design objects (page [69](#)).
15. Updating EAR files (page [70](#)).
16. Reporting customer modifications (page [71](#)).

The description of a task comprises an overview of the objects affected by the task, of the changes to be made to the affected objects in the target system before the upgrade, and of potential conflicts that can occur during the upgrade. For some tasks, further information is provided (e.g. technical remarks).

## Checking for Checked-out Objects

The **Check for Checked-out Objects** task checks if there are any checked-out objects in the target system. If an object is checked-out, e.g. in Process Designer, it may not be updated correctly.

This task is only performed in **Stage 3** (Checking Target System for Conflicts) for all objects exported from the target system that can be checked-out. For the checked-out objects, **TaskResultItem** elements are created in the stage result report.

Example report:

```
<TaskResult startTime="2012-08-24T09:02:34.964+02:00" endTime=
  "2012-08-24T09:03:39.583+02:00" taskClassname=
  "com.rockwell.mes.update.tasks.general.CheckCheckedOutObjects"
  hasErrors="true">
  <TaskResultItem affectedObject="DeAnzaForm" changeLevel="ObjectType"
    changeType="Other">
    <TaskResultItem affectedObject="ApplicationStart" changeLevel="Object"
      changeType="Other">
      <Error>The object is checked out on the target system.</Error>
    </TaskResultItem>
  </TaskResultItem>
</TaskResult>
```

## Updating UDA Definitions

The **Update UDA Definitions** task performs an update of **UDA Definition** objects.

If **Stage 2** (Computing Planned Changes) results in **Overwrite** (see "General Update Strategy" (page 47)), for each **A B \*** case (08 - 11) the result list will be expanded by an additional list covering the planned changes on UDA definition item level. The detail level is also available in **Stage 3** (Checking Target System for Conflicts).

Generally, this task does not delete UDA definitions and UDA definition items (for details, see "Technical Remarks" (page 52)).

### AFFECTED OBJECTS

The following objects are affected:

- UDA definitions and their dependent UDA definition items.

## ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- Adding new UDA definitions.
- Adding, modifying, and deleting UDA definition items related to the newly added UDA definition (custom UDA definition).
- Adding new UDA definition items to existing UDA definitions (non-custom UDA definition).

## CONFLICTS

See "How to Resolve Conflicts" (page [49](#)).

## TECHNICAL REMARKS

- To add, delete, or overwrite UDA definitions and UDA definition items, the task uses the import functionality of Process Designer. The following restrictions are known and apply to an update performed by this task:
  - Generally, this task does not delete UDA definitions and UDA definition items.  
However, the results of **Stage 2** (Computing Planned Changes) and **Stage 3** (Checking Target System for Conflicts) list the planned deletion operations, along with the information that no deletion will be performed.
  - Modifying the text length of a UDA definition item of the **String** type is allowed. However, a reduction of the text length is ignored. The results of **Stage 2** (Computing Planned Changes) and **Stage 3** (Checking Target System for Conflicts) list this fact.
  - Modifying the name of a UDA definition item is not allowed.
  - Modifying the type of a UDA definition item is not allowed.

## Updating Library Objects

The **Update Library Objects** task performs an update of **Library** objects. A library consists of two artifacts, the XML description file and the actual library binary, e.g. a JAR file. Libraries are not merged, they are either added, overwritten, or deleted.

### CONFIGURATION

The task configuration allows to exclude files from being processed by means of the Exclude element. Files can either be XML description files or the actual binary JAR files.

- To exclude a library, you need to indicate its XML description and binary JAR file. Add **Exclude** elements to the configuration, with the names of the files to be excluded.

Example: To exclude the *jgraph-5.12.4.0* library with its *jgraph-5.12.4.0.jar.xml* XML file and *jgraph-5.12.4.0.jar* binary attachment, add the following lines:

```
<TaskConfig classname="com.rockwell.mes.update.tasks.specific.UpdateLibraries">
  <Exclude>jgraph-5.12.4.0.jar.xml</Exclude>
  <Exclude>jgraph-5.12.4.0.jar</Exclude>
</TaskConfig>
```

### AFFECTED OBJECTS

The following objects are affected:

- XML files in the *Libraries* subdirectory.
- The corresponding library binaries in the *addon\_jars* subdirectory.

### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- None.  
Merging is not included in the task.

### CONFLICTS

See "How to Resolve Conflicts" (page 49).

## Updating DSX Objects (Generic DSX)

The **Update DSX Objects** task performs an update of some DSX object types like forms and activity sets. It does not perform merges of any kind.

There are objects that cannot be deleted since they are referenced in the target system of the customer. In this case, the task reports a warning. This situation is not considered as an error condition.

## CONFIGURATION

The following configuration applies to the task:

```
<TaskConfig classname="com.rockwell.mes.update.tasks.general.UpdateGenericDSX">
  <Exclude>UDADefinition</Exclude>
  <Exclude>Application</Exclude>
  <Exclude>ATDefinition</Exclude>
  <Exclude>ATRow</Exclude>
  <Exclude>DsMessages</Exclude>
  <Exclude>Libraries</Exclude>
  <Exclude>addon_jars</Exclude>
  <Exclude>DsImage</Exclude>
  <Exclude>image_attachments</Exclude>
  <Exclude>VPVersionConfiguration</Exclude>
  <Exclude>FSMConfiguration</Exclude>
  <Exclude>FlexibleStateModel</Exclude>
  <Exclude>SemanticPropertySet</Exclude>
  <Exclude>SemanticProperty</Exclude>
  <Exclude>ReportDesign</Exclude>
</TaskConfig>
```

The task supports two configuration options. It allows to exclude object types or objects by means of the **Exclude** element.

- To exclude an object type, add an **Exclude** element to the configuration, with the name of the directory containing the object type to be excluded.

Example: To exclude "ATRow", add the following line:

```
<Exclude>ATRow</Exclude>
```

- To exclude an object, add an **Exclude** element to the configuration, with the name of the directory containing the object and the name of the object file separated by a forward slash.

Example: To exclude "User pecadmin", add the following line:

```
<Exclude>User/pecadmin.xml</Exclude>
```

## MAPPING OF DIRECTORY NAMES

The task relies on a mapping of directory names to object types (**DKeyed** classes). The convention is to prefix the directory name with a **D**, e.g. the data object of **Part** is **DPart**. Unfortunately, the convention is broken for a couple of types, e.g. **DeAnzaForm** which maps to **DForm**. Some of the deviations are already handled in the task. If you need to introduce a new type that does not comply with the naming convention, you can add the mapping of the new type to the configuration by means of the **Property** element.



Example: Add the following element to the configuration to add its properties to the internal mapping

```
<Property key="DeAnzaForm" value="DForm" />
```

The following object types are already handled by the mapping:

- Activities, DAddOn
- DeAnzaForm, DForm
- DsImage, DGeneric
- DsList, DList
- DsMessages, DMessagePack
- EventSheetHolder, DForm
- Libraries, DAddOn
- Operation, DRouteOperation
- Subroutine, DLibrary

#### AFFECTED OBJECTS

The following objects are affected:

- AccessPrivilege
- Activities
- Application  
They are excluded from the **Update DSX Objects** task and processed by the **Update Application Objects** task (page 68).
- ATDefinition  
They are excluded from the **Update DSX Objects** task and processed by the **Update AT Definitions** task (page 57).
- DataDictionaryClass
- DeAnzaForm
- DsList
- EquipmentClass
- EventSheetHolder
- FSMConfiguration  
They are excluded from the **Update DSX Objects** task and processed by the **Update FSM Objects** task (page 62).
- Locale
- NamedFilter

- Operation
- ProductionLine
- ReportDataDefinition
- ReportDesign  
They are excluded from the **Update DSX Objects** task and processed by the **Update Report Design Objects** task (page [69](#)).
- SemanticProperty  
They are excluded from the **Update DSX Objects** task and processed by the **Update FSM Objects** task (page [62](#)).
- UDADefinition  
They are excluded from the **Update DSX Objects** task and processed by the **Update UDA Definitions** task (page [51](#)).
- User
- UserGroup
- VPVersionConfiguration  
They are excluded from the **Update DSX Objects** task and processed by the **Update Versioning Configuration Objects** task (page [68](#)).

#### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- None.  
Merging is not included in the task.

#### CONFLICTS

See "How to Resolve Conflicts" (page [49](#)).

#### TECHNICAL REMARKS

- For **Activity** objects that directly reference jar files (e.g. EIG-related activities), conflicts within the jar files are not detected. This means that the task detects if an **Activity** object was modified on the target system, but it cannot detect if only the jar file was modified. There are no **Activity** objects that directly reference jar files in the framework, so this limitation only affects customer-specific objects. Please consider this behavior when you migrate **Activity** objects of this kind to make sure there are no undetected conflicts.
- To add or overwrite objects, the task uses the import functionality of Process Designer. The following restrictions are known and apply to an update performed by this task:

- The internal GUID attribute of **Activity** and **Activity set** objects cannot be changed in case of overwriting. Since the attribute is not visible or editable in Process Designer, it must not be changed in the XML structure.
- For the deletion of objects, the same restrictions apply as in Process Designer.
- **User** objects cannot be deleted if the user deletion is prohibited globally. This is the default setting for PharmaSuite.

## Updating AT Definitions

The **Update AT Definitions** task performs an update of **AT Definition** objects. AT definitions are definitions for application tables whose data is stored in AT rows (see "Updating ATRow Objects" (page 59)).

### TIP

We recommend to execute the Update AT Definition task before the Update AT Row task.

## AFFECTED OBJECTS

The following objects are affected:

- AT definitions and their dependent objects:
  - AT column definition
  - AT index definition
  - Parameter

## ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- Adding AT column definitions, AT index definitions, and parameters to an existing AT definition.
- Deleting AT column definitions, AT index definitions, and parameters from an existing AT definition.

## CONFLICTS

See "How to Resolve Conflicts" (page 49).

## TECHNICAL REMARKS

- To add, delete, or overwrite AT definitions, the task uses the import functionality of Process Designer. The following restrictions are known and apply to an update performed by this task:
  - Deletion operations for AT column definitions and parameters from an AT definition indicated in **Stage 2** (Computing Planned Changes) are not performed in **Stage 4** (Performing Update on Target System). In the result list, the planned deletion operations are extended with the information that no deletion is performed, since AT rows may hold data that is used by existing AT definitions.  
Deletion operations for AT index definitions are performed.
  - If, in the target system, an AT index definition and parameters were deleted from an AT definition and the deleted items are still available in the new version, this is indicated as conflict in **Stage 3** (Checking Target System for Conflicts). Otherwise, the items would be added in **Stage 4** (Performing Update on Target System).
  - Adding AT column definitions, AT index definitions, and parameters to an AT definition in the target system is allowed. This is detected in **Stage 3** (Checking Target System for Conflicts). Due to technical reasons, AT index definitions and parameters within an AT definition of the target system are temporarily removed but recreated during **Stage 4** (Performing Update on Target System). Items to be recreated in Stage 4 are marked with the **Other** type in the result report of Stage 3.
  - The list of route steps in a parameter is not supported. References to route steps in a parameter recreated in **Stage 4** (Performing Update on Target System), are lost and must be reentered manually after the update.
  - AT column definitions are not deleted at all.
  - AT definitions with other AT definition as dependent objects cannot be merged automatically. This situation is reported as conflict in **Stage 2** (Computing Planned Changes).

### IMPORTANT

AT index definitions result in database table indexes. Thus, the restrictions of the underlying database for defining table indexes apply. You cannot define two indexes on the same table column or on the same column combinations in the same order.

The update task reports an error, if merging the AT index definitions would result in table index conflicts.

## Updating ATRow Objects

The **Update ATRow Objects** task performs an update of **ATRow** objects like choice lists, choice list elements, GHS statements, etc.

### CONFIGURATION

The task supports two configuration options. It allows to exclude AT row types or objects by means of the **Exclude** element.

- To exclude an AT row type, add an **Exclude** element to the configuration, with the name of the directory containing the AT row type to be excluded.  
Example: To exclude "X\_GHSStatement", add the following lines:

```
<TaskConfig classname="com.rockwell.mes.update.tasks.specific.UpdateATRow">
  <Exclude>X_GHSStatement</Exclude>
</TaskConfig>
```

- To exclude an object of an AT row type, add an **Exclude** element to the configuration, with the name of the directory containing the AT row type and the name of the object file separated by a forward slash.  
Example: To exclude "X\_GHSStatement/H200.xml", add the following lines:

```
<TaskConfig classname="com.rockwell.mes.update.tasks.specific.UpdateATRow">
  <Exclude>X_GHSStatement/H200.xml</Exclude>
</TaskConfig>
```

### AFFECTED OBJECTS

The objects in the following subdirectories of the *ATRow* directory are affected:

- X\_ChoiceList
- X\_ChoiceElement
- X\_RSHint
- X\_GHSStatement
- X\_AdministrativeTask
- X\_UFDefaultFilterProps
- X\_PhaseLib
- X\_UiUsecaseConfiguration

### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- Adding ATRow data cells to an existing ATRow.
- Deleting ATRow data cells from an existing ATRow.

## CONFLICTS

See "How to Resolve Conflicts" (page 49).

## TECHNICAL REMARKS

- The name of the XML file of an AT row object can differ from the name of the object due to operating system restrictions.
  - Example: Curly brackets (braces, {, }) are mapped to underscores (\_).
- The name of an AT row object in the reporting XML files consists of the name of the subdirectory, a forward slash, and the DSX object name of the AT row.
  - Example: objectName = *X\_GHSSStatement/H200*, xmlFileName = *H200.xml* in *X\_GHSSStatement* subdirectory
- To add, delete, or overwrite ATRow objects, the task uses the import functionality of Process Designer. The following restrictions are known and apply to an update performed by this task:
  - Deletion operations for AT data cells from an ATRow indicated in **Stage 2** (Computing Planned Changes) are not performed in **Stage 4** (Performing Update on Target System). In the result list, the planned deletion operations are extended with the information that no deletion is performed, since the data cells of the AT rows are defined by the data columns in the corresponding AT definitions.
  - Adding ATRow data cells to an AT row in the target system is allowed. This is detected in **Stage 3** (Checking Target System for Conflicts). These items are marked with the **Other** type in the result report of Stage 3.
  - ATRow data cells are not deleted at all.
  - ATRow objects that have parameters cannot be merged automatically. This situation is reported as conflict in **Stage 2** (Computing Planned Changes).
  - AT rows with other AT rows as dependent objects cannot be merged automatically. This situation is reported as conflict in **Stage 2** (Computing Planned Changes).

## Updating Access Privilege Objects

The **Update Access Privilege Objects** task performs an update of **Access Privilege** objects. It does not perform merges of any kind.

In **Stage 3** (Checking Target System for Conflicts), the task retrieves the **effectivityStart** property of all access privileges that will be migrated and sets it in both the BASE version and the NEW version of the access privilege. This means that a difference in the **effectivityStart** property will not cause conflicts and the target value will be kept after importing the NEW version.

### AFFECTED OBJECTS

The following objects are affected:

- Access Privilege objects.

### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- None.  
Merging is not included in the task.

### CONFLICTS

See "How to Resolve Conflicts" (page 49).

## Updating Image Objects

The **Update Image Objects** task performs an update of images, i.e. DSX objects of **Images** type (=DsImages, see "Updating DSX Objects (Generic DSX)" (page 53)). An image consists of two artifacts, the XML description file and the actual image binary. Images are not merged, they are either added, overwritten, or deleted.

### CONFIGURATION

The task configuration allows to exclude files from being processed by means of the **Exclude** element. Files can either be XML description files or the actual image binary files.

- To exclude an image, you need to indicate its XML description or binary image file. Add **Exclude** elements to the configuration, with the names of the files to be excluded.

Example: To exclude the *Access\_rights\_16* image with its *Access\_rights\_16.xml* XML file and *Access\_rights\_16.png* image attachment, add the following lines:

```
<TaskConfig classname="com.rockwell.mes.update.tasks.specific.UpdateImages">
  <Exclude>Access_rights_16.xml</Exclude>
  <Exclude>Access_rights_16.png</Exclude>
</TaskConfig>
```

#### AFFECTED OBJECTS

The following objects are affected:

- XML files in the *DSImage* subdirectory.
- The corresponding image binaries in the *image\_attachment* subdirectory.

#### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- None.  
Merging is not included in the task.

#### CONFLICTS

See "How to Resolve Conflicts" (page 49).

### Updating FSM Objects

The **Update FSM Objects** task performs an update of several objects that are related to Flexible State Models. Due to dependencies between the various objects, the individual updates must be performed in a pre-defined order.

#### AFFECTED OBJECTS

The following objects are affected:

- Flexible state model objects and their dependent objects:
  - State
  - Transition
  - State exit expression
  - Transition instance
  - Propagation definition
  - Semantic property override
- Semantic property objects
- Semantic property set objects
- FSM configurations and their dependent objects:
  - FSM configuration items



## UPDATE AND COMPARISON STRATEGY

For flexible state models, semantic property sets, semantic properties, and FSM configurations, the update strategy basically follows the general update strategy (page 47). The objects are compared on **attribute level**. This affects cases 08 - 11; shown in the table below. Differences from the general strategy are formatted **bold**. Additionally, slightly different rules apply for conflict detection.

Since comparison happens on attribute level, allowed changes as well as conflict detection rules are defined on attribute level as well.

Case	BASE	NEW	TARGET	Result of Stage 2	Result of Stage 3
8	A	B	B	<b>Merge/Conflict</b>	None
9	A	B	A	<b>Merge/Conflict</b>	Replace
10	A	B	C	<b>Merge/Conflict</b>	<b>Merge/Conflict</b>
11	A	B	---	<b>Merge/Conflict</b>	Conflict

Conflicts can occur in stage 2 if there are structural differences in the FSM objects between the **BASE** version and the **NEW** version (for details see "Conflicts (page 65)"). In stage 3, it is possible to merge on attribute level. Conflicts on attribute level, however, can occur as well.

## UNALLOWED CHANGES

### TIPS

The terms **set of state**, **set of transition**, **set of transition instance**, **set of propagation definition**, and **set of semantic property override** refer to collections of objects.

States and transitions are identified by their names.

Transition instances are identified by the triplet of state name, transition name, and target state.

Propagation definitions are identified by the tuple of objectType, fsmRelationshipName, expression, transition, and ignoreIfTransitionNameDoesNotExist. Semantic property overrides are identified by the pair of dataType and overrideDataValue.

Changing a set means adding an element to or removing an element from the set.

Changing an element within a set, e.g. nameMessageid with a state, does not imply a change of the set.

Changing an attribute on the target system leads to conflicts if the attribute was modified on the new system in a way that differs from the target change. This applies to the following attributes:

- **Flexible state model:** category, description, defaultState, failOnNullTransitionInstance, loggingEnabled, messagePackKey, set of state, set of transition, set of transition instance
- **State:** nameMessageId, state exit expression, set of semantic property override
- **Transition:** nameMessageId
- **State exit expression:** expression, falseTransitionName, trueTransitionName  
A state exit expression is a compound object but treated as single attribute. Therefore modifications on more than one attribute may lead to conflicts on the compound object.
- **Transition instance:** set of propagation definition, set of semantic property override
- **Semantic property:** category, description, defaultDataValue, semanticType
- **Semantic property set:** category, description
- **FSM configuration:** category, description

**Structural changes** within a flexible state model are generally not allowed. Changing the structure of an FSM that is in use might impact active or completed objects it governs, such as batches or orders. This might even cause an active object to have a state that does no longer exist. A change of this kind effectively represents a new FSM and would thus require a new FSM configuration. This applies not only to changes in the target system but also to changes between the **BASE** version and the **NEW** version.

Flexible state models are structurally changed if at least one of the following attributes/collections is changed:

- default state
- set of states
- set of transitions
- set of transition instances

## RELEASE-SPECIFIC UPDATE TASK

With PharmaSuite 8.4, a structural change of the **S88EquipmentClassStatusGraph** FSM was necessary due to the introduction of the equipment mass change feature. The **Approved-returnToDraft** transition instance was added to the FSM. In order to add the new transition instance to the existing FSM during a system update, the **UpdateFSMFrom83To84** task was added to the update-specific configuration file (*updateConfiguration.xml*). This task is similar to the standard **UpdateFSM** task.

## ALLOWED CHANGES

The following changes are allowed to be made on the following attributes of the original objects of the target system:

- **Flexible state model:** Adding and removing semantic property sets.
  - **State:** Adding and removing semantic properties and semantic property value overrides.
  - **Transition instance:** Adding and removing semantic properties and semantic property value overrides.
- **Semantic property set:** Adding and removing semantic properties.
- **FSM configuration:** Adding and removing FSM configuration items.
  - **FSM configuration item:** Modifying the **FSMRelationshipType** attribute.

## CONFLICTS

See "How to Resolve Conflicts" (page 49).

To resolve a **structural conflict** found for a flexible state model, we recommend to proceed as follows:

1. Create a new FSM with the desired structure.
2. Change the relationship type of the (old) existing FSM configuration items from **Automatic** to **Manual**.
3. Create new FSM configuration items and set their relationship type to **Automatic**, if desired.

With this approach, it is possible to keep existing objects alive according to their life-cycle model, while new objects can follow a different life cycle. It is also possible to track, when the different life cycles are used. Structural changes usually imply changes within the business logic and may require code changes.

**TIP**

Please keep in mind that having an old and a new version of an FSM in the system, may require the business logic (code) to support both versions simultaneously.

## UNSUPPORTED FEATURES

The update task does not support the following features:

- Parameters and UDAs defined for semantic property sets or flexible state models.
- State transition attribute definitions associated with states of flexible state models.
- Permission-related semantic properties added by FactoryTalk ProductionCentre are not populated as semantic property overrides to the states of the related FSM.

## TECHNICAL REMARKS

- Flexible state models
  - Deleting a semantic property association from a state or a transition instance also deletes an existing semantic property override, without further notification.
- Semantic property sets
  - Deleting a semantic property from a semantic property set may cause undetected data inconsistencies, if the property is associated to a state or transition instance of an FSM while within the FSM, the semantic property is no longer linked to the semantic property set that is associated with the FSM.
- FSM configuration objects
  - Deletion operations indicated in **Stage 3** (Checking Target System for Conflicts) are not performed in **Stage 4** (Performing Update on Target System). Instead, the **FSMRelationshipType** attribute is set to **Manual** for all **FSM configuration item** objects of the **FSM configuration** object.
- FSM configuration item objects
  - Deletion operations indicated in **Stage 3** (Checking Target System for Conflicts) are not performed in **Stage 4** (Performing Update on Target System). Instead, the **FSMRelationshipType** attribute is set to **Manual**.
  - Modifying the **object-type** attribute is not allowed.
  - For any given **FSM relationship** only one **FSM configuration item** with the **FSMRelationshipType Automatic** must exist. There are several scenarios, where potential conflicts are detected in **Stage 4** (Performing Update on Target System).

- The **FSMStateName** or **FSMRelationshipName** attributes define the identifiers for **FSM configuration item** objects. A slash (/) in these attributes is replaced by a double-slash (//) in the resulting XML structure.

## Updating Message Objects

The **Update Message Objects** task performs an update of Messages, i.e. DSX objects of **Messages** type (=DsMessages, see "Updating DSX Objects (Generic DSX)" (page 53)).

### TIP

In the following, we use the term **Message pack** instead of **Message** to denote the DSX object in order to be able to distinguish between the DSX object and the Message itself.

The task starts with a detailed analysis of each Message pack to detect conflicts and automatically merge most changes related to Message IDs, Messages, and localized Messages.

### IMPORTANT

Only Messages with the **en\_US** Locale are merged into the target system, if merging is needed to solve a conflict.

## AFFECTED OBJECTS

The following objects are affected:

- Message packs and their dependent objects
  - Message ID
  - Message: default content with **en\_US** Locale
  - Localized Messages with other Locales

## ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- Modifying existing Messages (of same Message ID) within an existing Message pack in order to provide localizations.
- Adding new Message packs.

## CONFLICTS

See "How to Resolve Conflicts" (page 49).

## Updating Application Objects

The **Update Application Objects** task performs an update of **Application** objects. It does not perform merges of any kind.

In **Stage 3** (Checking Target System for Conflicts), the task processes the **DefaultConfiguration** object as follows: It retrieves the value of the **MessageBrokerURL** configuration key from the target system and sets it in both the BASE version and the NEW version. This means that the target value will be kept after importing the NEW version.

### AFFECTED OBJECTS

The following objects are affected:

- Application objects

### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- None.  
Merging is not included in the task.

### CONFLICTS

See "How to Resolve Conflicts" (page 49).

## Updating Versioning Configuration Objects

The **Update Versioning Configuration Objects** task performs an update of the versioning configuration objects.

In general, the update strategy follows the general update strategy (page 47). Particularly, no merges are supported if **Stage 2** (Computing Planned Changes) results in **Overwrite**. This does not apply to deletion operations. They are not performed at all for the versioning configuration objects.

### AFFECTED OBJECTS

The following objects are affected:

- VPVersionConfiguration objects.

### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- None.  
In fact, FactoryTalk ProductionCentre does not allow to add or delete versioning configuration objects.

**CONFLICTS**

See "How to Resolve Conflicts" (page 49).

**TECHNICAL REMARKS**

- To add or overwrite versioning configurations, the task uses the import functionality of Process Designer.
- Generally, this task does not delete versioning configurations. However, the results of **Stage 2** (Computing Planned Changes) and **Stage 3** (Checking Target System for Conflicts) list the planned deletion operation along with the information that no deletion will be performed.

**Updating Report Design Objects**

The **Update Report Design Objects** task performs an update of **Report Design** objects.

In general, the update strategy follows the general update strategy (page 47). Particularly, no merges are supported if **Stage 2** (Computing Planned Changes) results in **Overwrite**.

**AFFECTED OBJECTS**

The following objects are affected:

- Report Design objects.
- Compiled report design ATRow objects in the **X\_ReportCompiled** application table.

**ALLOWED CHANGES**

The following changes are allowed to be made to the original objects of the target system:

- None.

**CONFLICTS**

See "How to Resolve Conflicts" (page 49).

**TECHNICAL REMARKS**

- This task also handles the compiled report design belonging to each report design.
  - When a report design was added or overwritten, the compiled report design is generated.
  - Before a report design is deleted, the compiled report design is deleted.
- To add or overwrite report designs, the task uses the import functionality of Process Designer.

## Updating EAR Files

The **Update EAR Files** task deploys new or changed EAR files to the application server and undeploys EAR files that are no longer used. This applies only to the primary application server.

The EAR deployment directory can be specified during the update process.

### AFFECTED OBJECTS

The following objects are affected:

- The PharmaSuite help EAR file.
- Any EAR file located in the *EAR\_FILES* directory of the upgrade installer.

### ALLOWED CHANGES

The following changes are allowed to be made to the original objects of the target system:

- None.  
Merging is not included in the task.

### CONFLICTS

See "How to Resolve Conflicts" (page 49).

### TECHNICAL REMARKS

- If you run several application servers in your PharmaSuite environment, each application server beyond the primary application server must be updated manually. For this purpose, proceed as follows:
  1. Copy all new and overwritten EAR files from the deployment directory of the primary application server to the deployment directory of each of the other application servers.
  2. Delete all deleted EAR files from the deployment directory of each of the other application servers.
  3. The list of new, overwritten, or deleted EAR files is provided in the result file of **Stage 4** (Performing Update on Target System):  
*UPDATE\_TARGET\_SYSTEM.xml*.
- If the update process is not executed directly on the application server, you can also use a remote directory for the EAR deployment directory.  
Please ensure that the user who performs the update has read and write access to the remote directory.



## Reporting Customer Modifications

The **Report Customer Modifications** task creates a report of the objects that have been added to, modified in, or deleted from the target system before the update. This applies only to the DSX object types used by PharmaSuite.

Usually, there are many **Part** (material) objects and **User** objects available in a customer system which would be considered as modifications and thus be reported. To avoid extensive reports, **Part** (material) objects and **User** objects are excluded by default.

This task is only performed in **Stage 3** (Checking Target System for Conflicts) and generates the *CustomerSystemModificationAnalysis.xml* report file in the update installation directory.

The report contains three main **TaskResultItem** elements with the following values for the **affectedObject** attribute:

- **AdditionalArtifacts**  
contains information on all objects that have been added to the target system.
- **ModifiedArtifacts**  
contains information on all objects that have been modified on the target system.
- **DeletedArtifacts**  
contains information on all objects that have been deleted from the target system.

Each of the main **TaskResultItem** elements contains nested **TaskResultItem** elements for each object type that has been modified. The object type elements contain **TaskResultItem** elements with the file names of the objects that were added, modified, or deleted.

For the ATRow objects, there are three levels of nested elements: for the ATRow object type, for the subdirectory containing the changes, and for the modified file. If an entire subdirectory is missing in the **BASE** version, the contents of the corresponding target system directory are listed as added and an additional **Info** element is added to provide the information that the entire subdirectory is missing in the **BASE** version. The same applies if a subdirectory available in the **BASE** version is missing in the **TARGET** version.

### Example report:

```
<xml-fragment startTime="2012-04-19T16:24:07.631+02:00" taskClassname=
    "com.rockwell.mes.update.tasks.general.ReportCustomerModifications"
    hasErrors="false" endTime="2012-04-19T16:24:07.677+02:00">
  <TaskResultItem affectedObject="AdditionalArtifacts" changeLevel="System"
    changeType="Add" xmlns="http://rockwell.com/mes/update/databeans">
    <TaskResultItem affectedObject="DeAnzaForm" changeLevel="ObjectType"
      changeType="Add">
      <TaskResultItem affectedObject="form_xC.xml" changeLevel="Object"
        changeType="Add"/>
    </TaskResultItem>
  </TaskResultItem>
  <TaskResultItem affectedObject="ModifiedArtifacts" changeLevel="System"
    changeType="Overwrite" xmlns="http://rockwell.com/mes/update/databeans">
    <TaskResultItem affectedObject="ATRow" changeLevel="ObjectType"
      changeType="Overwrite">
      <TaskResultItem affectedObject="X_AdministrativeTask" changeLevel="ObjectType"
        changeType="Overwrite">
      <TaskResultItem affectedObject="LockingManagement.xml" changeLevel="Object"
        changeType="Overwrite"/>
      </TaskResultItem>
    </TaskResultItem>
  </TaskResultItem>
  <TaskResultItem affectedObject="DeletedArtifacts" changeLevel="System"
    changeType="Delete" xmlns="http://rockwell.com/mes/update/databeans">
    <TaskResultItem affectedObject="ATRow" changeLevel="ObjectType" changeType="Delete">
      <TaskResultItem affectedObject="X_PhaseLib" changeLevel="ObjectType"
        changeType="Delete">
        <Info>The directory does not exist in the target version.</Info>
        <TaskResultItem affectedObject="_067C1850-78C5-ACB6-441F-388E88A8EB84_.xml"
          changeLevel="Object" changeType="Delete"/>
      </TaskResultItem>
    </TaskResultItem>
    <TaskResultItem affectedObject="DeAnzaForm" changeLevel="ObjectType"
      changeType="Delete">
      <TaskResultItem affectedObject="formAx_.xml" changeLevel="Object"
        changeType="Delete"/>
    </TaskResultItem>
  </TaskResultItem>
</xml-fragment>
```

## Technical Details of a Data Migration

This section provides technical details of a data migration.

### Blockwise Update

A data migration task updates the data that resides in the newly updated PharmaSuite system. Depending on the object type it processes, a data migration task can modify a large number of data objects and thus may cause memory or timeout issues. In order to prevent such issues, data migration tasks allow a blockwise update. It is optional and should be used for data migration tasks that are expected to modify a large number of data objects. PharmaSuite supports the following types of blockwise update:

- **Update database entries**  
An SQL statement that is used to update database entries shall be configured to update only a limited number of rows at a time and it shall be executed multiple times until all rows have been updated.  
This configuration allows to prevent database transaction timeout issues.
- **Update database objects (load, update, save)**  
A task that is used to update database objects shall be configured to load only a limited number of objects at a time, update them, and save the objects. Then, the next groups of objects are processed until all objects have been updated.  
This configuration allows to prevent memory issues.

The following parameters are available to configure a blockwise update:

- *sqlUpdateBlockSize* used for SQL update statements.  
The default value is 100000.
- *loadObjectsBlockSize* used to load objects.  
The default value is 1000.

A blockwise update can be configured per task or globally for all data migration tasks:

- **Task-specific configuration**  
To change the block size to be used for a specific data migration task, modify its configuration in the migration-specific configuration file (*updateConfiguration.xml*).  
Example: You increase the number of equipment entities to be loaded by the **UpdateS88EquipmentBarcode82To83** task from 1000 (default) to 5000.

```
<TaskConfig classname=
  "com.rockwell.mes.update.tasks.onetime.UpdateS88EquipmentBarcode82To83" >
  <loadObjectsBlockSize>5000</loadObjectsBlockSize>
</TaskConfig>
```

#### ■ Global configuration

To change the block size for all data migration tasks, run the data migration wizard with an additional VM argument as startup parameter.

Example: You reduce the number of data entries to be updated by an SQL statement for all tasks that use this mechanism from 100000 (default) to 50000 by setting `-DsqlUpdateBlockSize=50000` as startup parameter.

The task-specific configuration has priority over the global configuration. If neither is specified, the default block sizes are used.

Please keep in mind that there is a trade-off between performance and reliability: if you use smaller block sizes there is very low risk for memory or transaction timeout issues, but the update tasks will be slower. If you use larger block sizes, the tasks run faster, but too big blocks could lead to memory or transaction timeout issues.

To see if a specific data migration task uses a blockwise update and if so, which type, check the **Blockwise Update** section in the description of each data migration task.

## Migration Tasks

The migration process performs the following migration tasks. Their sequence is defined in the migration-specific configuration file (*updateConfiguration.xml*).

1. Check prerequisites.  
Checks the general prerequisites for the migration process.  
Checks if the installed versions of PharmaSuite and FactoryTalk ProductionCentre match the expected versions.
2. Update transition conditions of activity sets (page [75](#)).
3. Update usage list-related tables (page [76](#)).

The description of a task comprises an overview of the objects affected by the task, of the changes to be made to the affected objects in the target system before the upgrade, and of potential conflicts that can occur during the upgrade.

## Migrating Transition Conditions of Activity Sets

The **UpdateTransitionConditions83To84** task updates the transition conditions of all activity sets related to master recipes, master workflows, and building blocks.

Prior to PharmaSuite 8.4, the default transition condition used for any activity set transition was *transitionUtil.inStepsFinished()*. It was prepended to any other transition condition set in Recipe and Workflow Designer or used by itself for non-conditional transitions. However, this condition evaluates to true if an incoming step is in the ERROR status, in this case it is typically not desired to activate the next step. With PharmaSuite 8.4, the default transition condition is *transitionUtil.inStepsFinished()&&!transitionUtil.inStepsError()*. The migration task updates all existing activity set transitions.

### AFFECTED OBJECTS

All activity set transition conditions that contain *transitionUtil.inStepsFinished()* but not *transitionUtil.inStepsFinished()&&!transitionUtil.inStepsError()* are affected.

The transition conditions containing Pnuts code (annotated with *@Pnuts*) are not migrated. If required, modify them manually according to the recommendation for Pnuts expressions. For details, see section "Editing Transition Conditions", chapter "Adapting the Workflows of the Production Execution Client" in Volume 1 of the "Technical Manual Configuration and Extension" [A11] (page 91)).

### ALLOWED CHANGES

No changes are anticipated.

### CONFLICTS

No conflicts are expected for this task.

### BLOCKWISE UPDATE

The task updates the database using an UPDATE SQL statement with a blockwise update. The *sqlUpdateBlockSize* property is used to determine the block size. Default is 100000.

## Migrating Usage List-related Tables

The **UpdateUsageListTables83To84** task clears the usage list-related tables (AT\_X\_CBBUsageIdxHeader and AT\_X\_CBBUsageIdxEntry) of a PharmaSuite 8.3 installation.

The introduction of the equipment class usage list in Data Manager made this change necessary.

### AFFECTED OBJECTS

With PharmaSuite 8.4, the AT\_X\_CBBUsageIdxHeader table became obsolete and was replaced by the AT\_X\_UsageIdxHeader table. The AT\_X\_EQMUsageIdxEntry table was introduced with PharmaSuite 8.4.

The system fills the usage list-related tables when the feature is used for the first time.

### ALLOWED CHANGES

No changes are anticipated.

### CONFLICTS

No conflicts are expected for this task.

### BLOCKWISE UPDATE

The task updates the database using a DELETE SQL statement with a blockwise delete. The *sqlUpdateBlockSize* property is used to determine the block size. Default is 100000.

## Adapting the Upgrade Installer

This section provides information how to adapt the upgrade installer for your specific purposes. There are several sources of information you can use to identify the necessary changes. Usually, the results of **Stage 3** (Checking Target System for Conflicts) provide an overview of the artifacts with conflicts (page 47). In addition to the conflict artifacts, there are other circumstances that necessitate adaptations to artifacts (page 78). Taken together, the indicated sources provide you with all the input you need to guide your adaptations.

Finally, the guidelines for implementing data migration tasks (page 85) provide useful information, especially for migrating mass data.

For the creation of new update or migration tasks (page 80), we recommend to work in Eclipse IDE which is supported by the UpdaterSDK.

To adapt the upgrade installer, perform the following steps:

1. Extract the Updater SDK.  
In the **Extraction Packages** step of the installation wizard, select the optional **SDK for the Update Package**.  
After the **Extraction** step has completed, click the **Cancel** button to close the installation wizard.
2. Create the workspace.  
Run the `<INSTALL_DIR>\UpdaterSDK\create_workspace.bat` batch file.
3. Resolve conflicts.  
Adapt the base version in `<INSTALL_DIR>\UpdaterSDK\src\main\resources\BASE_VERSION` and the new version in `<INSTALL_DIR>\UpdaterSDK\src\main\resources\NEW_VERSION`.  
For further details, see "Resolving DSX Conflicts" (page 78).
4. Optional: Add new upgrade task.  
For further details, see "Creating New Upgrade Tasks" (page 80).
5. Create the adapted installer.  
Run the `<INSTALL_DIR>\UpdaterSDK\build_installer.bat` batch file.  
The new installer will be written to `<INSTALL_DIR>\UpdaterSDK\CustomInstaller.jar`.

## Detecting Artifacts that Need to Be Adapted

Sometimes customized objects do not conflict with standard artifacts but are indirectly affected by API changes or structural changes within the new version of PharmaSuite.

- For structural changes, first, review the report of **Stage 3** (Checking Target System for Conflicts): `<INSTALL_DIR>/COMPUTE_PLANNED_CHANGES.xml`. Secondly, compare the `BASE_VERSION` and `NEW_VERSION` directories. We recommend to use a diff tool.
- For API changes, see the PharmaSuite Release Notes [C1] (page 92).

In addition to retrieving a list of customized artifacts from your source control system, we recommend to review the *CustomerSystemModificationAnalysis.xml* report that lists all customized objects (see "Reporting Customer Modifications" (page 71)). Thus, you get a complete overview of the changes you have implemented.

## Resolving DSX Conflicts

If **Stage 3** (Checking Target System for Conflicts) does not complete successfully, the reported conflicts need to be analyzed and resolved. For this purpose, review the stage-specific report: `<INSTALL_DIR>/COMPARE_WITH_TARGET_SYSTEM.xml`. In the report, conflicts are marked by the **Error** element (see "Reporting Update Results" (page 38)). When you scan the file for the `<Error>` string, you will quickly find the important parts of the report. The description provides a good hint for the cause of the conflict and how to resolve it. However, in some cases, the description is quite generic.

As a first step of the resolving process, you need to find out what kind of object is in conflict. The **affectedObject** attribute of the respective **TaskResultItem** element provides the type of the object and the object itself.

Example:

```
<TaskResultItem affectedObject="Subroutine" changeLevel="ObjectType" changeType="Merge">
  <Error>Conflicts detected.</Error>
<TaskResultItem affectedObject="wip_Support" changeLevel="Object"
  changeType="Overwrite">
  <Error>Cannot update object [wip_Support] on the target system. [wip_Support] is marked
    for update, but it was already modified on the target system.</Error>
</TaskResultItem>
```

As you can see in the example, the *wip\_Support* subroutine was modified in both the **NEW** version and the **TARGET** version. To resolve the conflict, the customizations done in the target system need to be merged into the subroutine of the **NEW** version. For further details, see "Merging DSX Artifacts" (page 79).



**IMPORTANT**

You need to assemble the XML files after you have changed them. In general, XML files and the DSX objects must be in sync.

Please refer to "How to Resolve Conflicts" (page 49) to understand when merging of DSX artifacts is required. For some conflicts, it is sufficient to do a rather simple adaption of the **BASE** and **NEW** version by replacing the XML file with the customer version.

### Merging DSX Artifacts

A merge of DSX artifacts is best done in Process Designer.

To merge DSX artifacts, proceed as follows:

1. Set up a development system based on the **NEW** version.
2. Identify an object you need to merge (as described in "Resolving DSX Conflicts" (page 78)).
3. In your **custom** development system, create a copy of the corresponding object and export it.
4. Start Process Designer and import the exported object into the **NEW** development system.
5. Merge your customizations into the imported object.
6. Export the customized object and run the  
`<INSTALL_DIR>\UpdaterSDK\disassembleDsxFile.bat` batch file to disassemble the object.

Depending on the amount of extensions and the number of relevant changes in the **NEW** version, other approaches may be preferable. So it may be less effort to start in Step 1 from the customized system and import new artifacts. Furthermore, there may be cases where it is not possible to import an artifact due to dependencies.

That is why you should handle the merging with care and decide from case to case which approach will fit best.

When you are finished with merging your customizations into the **NEW** system, the upgrade installer itself must be updated. Proceed as follows:

1. Copy the new XML file into the *NEW\_VERSION* folder in order to replace the existing one.
2. Copy the XML file of your original objects into the *BASE\_VERSION* folder in order to replace the existing one.

3. Important  
Assemble the **NEW** version and the **BASE** version.
4. Create a new installer (page 77) and run the new installer. The conflict will be resolved now.

For trivial modifications, consider merging the DSX artifacts directly on XML level. In this case, open the XML file and modify the attributes of the object accordingly. Make sure to retain the alphabetic sorting of the elements. A wrong sorting can lead to issues in **Stage 5** (Verifying Updated Target System) since the comparison will fail.

This approach works well for all of the DSX artifacts except for **Library** and **Image** objects. They require a bit more attention since they consist of a pair of files: an XML description file and the actual binary. The pairs are located in the *DsImage* and *image\_attachments* respectively *Libraries* and *addon\_jars* folders. You need to make sure to update such pairs consistently.

## Creating New Upgrade Tasks

Along with the installer comes an SDK, the UpdaterSDK, which supports you in the development of new upgrade tasks. The UpdaterSDK helps you to set up a development environment, build your task, and build unit tests. In case you have trouble to get your task up and running, there is also debugging support.

### DEVELOPMENT ENVIRONMENT

To set up the development environment, proceed as follows:

1. If not already done, run the  
`<INSTALL_DIR>/UpdaterSDK/create_workspace.bat` batch file.  
This will create an Eclipse project.
2. Start Eclipse with an empty workspace outside `<INSTALL_DIR>`.
3. Import the generated project from the `<INSTALL_DIR>/UpdaterSDK/customtask` directory.

### EXAMPLE TASKS

Example tasks are shipped with the UpdaterSDK to demonstrate what an upgrade task can look like. Run and debug the example tasks. Use the launch configuration delivered with the UpdaterSDK for this purpose. The examples also include a unit test for the task.

## NEW TASK

To develop and test a new task, proceed as follows:

1. In the *src* folder, create a package for the task.  
Example: *com.rockwell.mes.update.tasks.<yourTask>*
2. In the new package, create a new class implementing *IUpdateTask*.  
See "Useful Base Classes" (page 83) for some base classes that are of use to you.
3. Implement the required methods (see "Interface to Be Implemented" (page 82)).
4. Write a unit test for your task (see "Writing Unit Tests for Tasks" (page 85)).
5. If there is a need to debug your code, use the Eclipse launch configurations that support debugging.
6. Create a JAR that contains your new task (see "Creating a JAR from Eclipse" (page 85)).
7. Configure the new task in  
*<INSTALL\_DIR>\UpdaterSDK\src\main\resources\updateConfiguration.xml*  
configuration file.
8. Rebuild the updater.

### TIP

The upgrade engine creates a task instance using the default constructor. It is important to understand that the same instance of a task is used for the complete upgrade process.

## LOGGING

To configure the default log level for your new task, add it to the log4j configuration in  
*<INSTALL\_DIR>\UpdaterSDK\src\main\resources\log4jupdater.properties* (see "Logging" (page 42)).

## DEBUGGING

If you need to debug the upgrade process, please use the launch configuration that is already available in Eclipse.

## Interface to Be Implemented

All upgrade tasks must implement the *com.rockwell.mes.update.engine.interfaces.IUpdateTask* interface. It specifies the methods that are called by the upgrade engine. For each stage of the upgrade process, a method is available.

- **Stage 1** (Checking Update Configuration)/(Checking Configuration):  
*getListOfHandledObjects(final TaskConfig taskConfig, final ISystemDataAccess systemDataAccess)*  
 returns a list of all objects handled by this task.
- **Stage 2** (Computing Planned Changes):  
*computePlannedChanges(final TaskConfig taskConfig, final IUpdateContext updateContext)*  
 determines the modifications required to upgrade from the **BASE** version to the **NEW** version.  
 Returns a **TaskResult** XML element.
- **Stage 3** (Checking Target System for Conflicts):  
*compareWithTargetSystem(final TaskConfig taskConfig, final IUpdateContext updateContext)*  
 determines potential conflicts when upgrading the target system.  
 Returns a **TaskResult** XML element similar to Stage 2 but enriched with **<Error>** elements to report conflicts.
- **Stage 4** (Performing Update on Target System)/(Migrating Data on Target System):  
*updateTargetSystem(final TaskConfig taskConfig, final IUpdateContext updateContext)*  
 performs the upgrade on the target system.  
 Returns also a **TaskResult** XML element that may be enriched by additional errors or warnings in case of upgrade issues.
- **Stage 5** (Verifying Updated Target System)/(Verifying Migrated Target System):  
*verifyUpdatedSystem(final TaskConfig taskConfig, final IUpdateContext updateContext)*  
 verifies the success of the upgrade.  
 Returns a **TaskResult** element that will be empty in a perfect world.  
 Nevertheless, it may hold a couple of entries for objects that could not be deleted. These entries are allowed since objects with connected data cannot be deleted. If there are other entries, the upgrade was not successful.

## Useful Base Classes

The following base classes support you in implementing new upgrade tasks:

- *com.rockwell.mes.update.tasks.specific.ObjectTypeDSXTaskBase*  
is applicable for most of the DSX object types. The class is a good starting point to merge objects on attribute level. It implements *IUpdateTask* and handles most of the behavior a task is required to provide. The class already implements the complete logic on object level which is quite generic. There are only two abstract methods that need to be overwritten:
  - **Stage 2** (Computing Planned Changes)  
*computePlannedChangesOnAttributeLevel()*  
compares the attributes of an object.
  - **Stage 3** (Checking Target System for Conflicts)  
*compareBaseWithTargetOnAttributeLevel()*  
checks for conflicts on attribute level.
- *com.rockwell.mes.update.tasks.specific.ObjectTypeAndBinDSXTaskBase*  
derives from *ObjectTypeDSXTaskBase*. The class can be used to handle object types with binary attachments (e.g. **Library** and **Image** objects). It adds an abstract method:
  - *checkAttachmentConsistency()*  
checks the consistency of assigned XML and binary objects.

For more details, please refer to the Java Documentation provided with the pre-configured Eclipse project [C2] (page 92).

## Relevant APIs

Upgrade tasks need to access the middle tier and need to know about their context. The infrastructure access is encapsulated in the upgrade engine in order to have a **common implementation** for all tasks and to **increase the testability** of tasks.

The infrastructure-related support classes are located in the *com.rockwell.mes.update.engine.interfaces* package:

- The *IMiddletierAccess* class provides the following methods:
  - *String importObject(final DKeyed objectToImport, final List<DKeyed> allObjects)*  
imports an object into the target system.
  - *String deleteObject(final DKeyed objectToDelete)*  
deletes an object from target system.
  - *String deleteATRow(final DATRow atRowToDelete, final String atDefinitionName)*  
deletes an ATRow.

- *ServerInfo* *getServerInfo()*  
retrieves the server information.
- *IFunctionsEx* *getFunctions()*  
gets access to functions.
- The *ISystemDataAccess* class provides the following methods:
  - *List<DKeyed>* *getSystemData(final DataSourceType dataSourceType)*  
retrieves the data of the given source system/version.
  - *File* *getDataDirectory(final DataSourceType dataSourceType)*  
retrieves the root data directory of the data source.
  - *void* *setReexportAfterUpdate()*  
sets the trigger to re-export the target system for verification after the update.
  - *File* *getObjectXmlSourceDirectory(final DataSourceType dataSourceType)*  
retrieves the root directory of the disassembled DSX of the given source system/version that contains all data objects as XML source files.
- The *IDatabaseAccess* class provides the following methods:
  - *List<String[]>* *fetchData(String sqlStatement)*  
retrieves a list that contains the result rows of an SQL statement. The attributes of the row are returned as array of strings.
  - *int* *executeStatement(String sqlStatement)*  
support class to execute SQL statements on the database.

The *com.rockwell.mes.update.engine.taskhelper* package also offers helper classes:

- *DirectoryDiffAnalyzer*  
computes the planned changes by given *DirectoryDiff* and checks for conflicts with the target system.
- *DisassembledFilesHelper*  
provides common operations on disassembled files.
- *ExcludeConfigHelper*  
supports configuration of excludes using directory and file level.
- *SubdirectoriesDiff*  
compares two directories, which have one layer of subdirectories below.
- *TaskHelper*  
provides general task helper utility methods.
- *UpdateTaskResultBuilder*  
builds the result bean structures.

For more details, please refer to the Java Documentation provided with the pre-configured Eclipse project [C2] (page 92).

## Writing Unit Tests for Tasks

When the task interface was designed, testability was a very important criterion. In order to support the development of unit tests, mocks are provided for each infrastructure-related class (e.g. *UpdateContextMock*). For a unit test example, see the *com.rockwell.mes.update.tasks.examples.UpdateFormTests* class. The corresponding test data is located in *src/test/resources/UpdateFormTests*.

## Creating a JAR from Eclipse

To create a JAR from Eclipse, proceed as follows:

1. Right-click <your project>.
2. Select **Export/Java/JAR file**.
3. Select only **customtask/src/main/java** as resources to be exported.
4. Set the export destination to  
`<INSTALL_DIR>\UpdaterSDK\src\main\resources\jars\CustomTask.jar`.

## Guidelines for Implementing Data Migration Tasks

This section describes how to optimize data migration tasks for mass data migration. When migrating mass data the following negative side effects can occur: high memory consumption because a large set of data is processed and long-running tasks because a lot of data is analyzed and updated.

Thus optimization can be crucial to avoid OutOfMemory situations and also improve performance.

For a typical data migration task, **Stage 2** (Computing Planned Changes) does not yet retrieve data from the database, but usually just returns a static message that indicates the general purpose of the task, such as "All master recipes will be migrated to hold the Discrete flag, set to False."

Thus the large set of data to be processed becomes noticeable

- in **Stage 3** (Checking Target System for Conflicts), when the task
  - retrieves data from the database,
  - analyzes the data, if further analysis is required after loading,
  - maintains the list of objects that are to be migrated later,
- in **Stage 4** (Performing Update on Target System), when the task updates the data,
- and also when it processes the TaskResultItems of the stage result reports.

A general recommendation is to combine two or more tasks operating on the same structures. This approach allows to load each structure once and then analyze and/or update it for the different tasks.

The subsequent sections provide optimization means for the affected areas and an example migration task.

### Processing a List of TaskResultItems vs. Logging

To process a large number of objects and to create one **TaskResultItem** element or **info**(rmation) message element per process object in parallel can lead to high memory consumption because the latter occupies memory until the result is written at the end of a stage.

Consider to add a summary (e.g. 1,000,007 of 1,500,000 objects migrated successfully) in the **TaskResultItem** and to log the details in *update.log* or configure a dedicated log file for your task.

Example log:

```
### log UpdateSampleTask to file ###
log4j.appender.updateSampleTasklogfile=org.apache.log4j.RollingFileAppender
log4j.appender.updateSampleTasklogfile.File=${com.rockwell.mes.update.logpath}/
UpdateSampleTask.log
log4j.appender.updateSampleTasklogfile.MaxFileSize=5MB
log4j.appender.updateSampleTasklogfile.MaxBackupIndex=100000
log4j.appender.updateSampleTasklogfile.layout=org.apache.log4j.PatternLayout
log4j.appender.updateSampleTasklogfile.layout.ConversionPattern=%d{ISO8601} %5p
%c{1}\:%L - %m%n

# UpdateSampleTask logging
log4j.logger.<TASK_PACKAGE_NAME>=INFO, updateSampleTasklogfile
```

### Loading Objects to Be Migrated

The objects to be migrated are usually loaded in **Stage 3** (Checking Target System for Conflicts).

FactoryTalk ProductionCentre filters do not support paging. For this reason, SQL statements with FastBeanReader (e.g. *fetchDataPaged(IFastBeanReaderFilter<T>, int)*) or *PCCContext.getFunctions().getArrayDataFromActive(String)* are preferred. They load only the data required for the analysis instead of all object instances, which would consume more memory. Additionally, the list of objects to be migrated should contain only the minimum required data in simple data types, e.g. string, long, pair.

#### TIP

We recommend to use SQL statements to extract data from the database. However, for modifications use the API, if possible.



If the analysis requires loading complex structures, consider to first load a list of objects to analyze (e.g. list of object keys) and then process chunks of the list. The chunk size should be a reasonable compromise between performance and memory consumption.

The list of objects to be migrated should contain the minimum required data for **Stage 4** (Performing Update on Target System) in order to load only the objects that have to be updated. A list of object keys or pair objects requires less memory than a list of *IMESRtOperations*, for example, which may also hold additional references, e.g. *IMESOperation* and *IMESRtUnitProcedure*.

## Updating the Target System

**Stage 4** (Performing Update on Target System) usually operates on a structure rather than on a single object. The structures should be loaded either one by one (if they are big, e.g. a recipe structure) or in chunks. When the update of an object or chunk is completed, all references should be released, call *IRecipeStructureModel.destruct()*, for example. Also consider to clear FactoryTalk ProductionCentre caches of objects only used by the task, especially if cached objects hold other references which are no longer needed. This also applies to the loading of objects to be migrated if the analysis has loaded the objects into the site cache. Another option is to temporarily disable caching.

When designing the task keep in mind that other tasks usually keep a list of objects to be migrated. Your task should be able to share memory consumption with the other data migration tasks.

## Verifying the Updated System

The **Stage 5** (Verifying Updated Target System) implementation usually can reuse the SQL statements of **Stage 3** (Checking Target System for Conflicts) (section "Loading Objects to Be Migrated" (page 86)) and only checks that the list is empty. If not, some objects may not have been migrated successfully.

## Example of a Data Migration Task

```
/**
 * Sample update task example.
 */
public class UpdateSampleTask implements IUpdateTask {

    /** The logger */
    private static final Log LOGGER = LogFactory.getLog(UpdateSampleTask.class);

    /** SQL query to retrieve objects to update */
    private static final String SELECT_OBJECTS_IN_OLD_FORMAT = " FROM ..." + " WHERE...";

    /** column name */
    private static final String OBJECT_KEY = "...";

    /** column name */
    private static final String OBJECT_NAME = "...";

    /**
```

```

    * Stores the data of the objects that have to be migrated.
    * Each object is represented by a Pair:
    * first - key
    * second - name
    */
private List<Pair<Long, String>> objectsToUpdate;

@Override
public List<File> getListOfHandledObjects(TaskConfig taskConfig, ISystemDataAccess
                                         systemDataAccess) {
    // This task does not modify release data, so we simply return an empty collection
    return Collections.EMPTY_LIST;
}

@Override
public List<TaskResultItem> computePlannedChanges(TaskConfig taskConfig, IUpdateContext
                                                  updateContext) {
    TaskResultItem resultItem =
        createTaskResultItem(getMessage("TaskPlannedChanges_AffectedObjects"));
    return Collections.singletonList(resultItem);
}

@Override
public List<TaskResultItem> compareWithTargetSystem(TaskConfig taskConfig,
                                                    IUpdateContext updateContext) {
    // Analyze data and load as less as possible information needed for
    // the next stage
    objectsToUpdate = loadObjectsToMigrate();

    // Summary message
    String generalMessage = String.format(getMessage("TaskCompareTarget_Message"),
                                           objectsToUpdate.size());
    final TaskResultItem resultItem = createTaskResultItem(generalMessage);
    LOGGER.info(generalMessage);
    for (final Pair<Long, String> objectToUpdate : objectsToUpdate) {
        LOGGER.info(objectToUpdate.getSecond());
    }
    return Collections.singletonList(resultItem);
}

@Override
public List<TaskResultItem> updateTargetSystem(TaskConfig taskConfig, IUpdateContext
                                              updateContext) {
    // Summary message
    String generalMessage = getMessage("TaskUpdateTarget_Message");
    final TaskResultItem resultItem = createTaskResultItem(generalMessage);
    LOGGER.info(generalMessage);

    for (final Pair<Long, String> objectToUpdate : objectsToUpdate) {
        updateObjectInTransaction(objectToUpdate, resultItem);
    }
    // Clean up cache after the task completes.
    // Cache size should be configured reasonably, e.g. 100,
    // otherwise other tasks may have OutOfMemory
    clearCaches();
    return Collections.singletonList(resultItem);
}

@Override
public List<TaskResultItem> verifyUpdatedSystem(TaskConfig taskConfig, IUpdateContext
                                              updateContext) {
    objectsToUpdate = loadObjectsToMigrate();
}

```

```

TaskResultItem resultItem;
if (!objectsToUpdate.isEmpty()) {
    resultItem = createTaskResultItem(getMessage("TaskVerify_Error_Message"));
    LOGGER.info(getMessage("TaskVerify_Error_Message"));
    for (final Pair<Long, String> objectToUpdate : objectsToUpdate) {
        resultItem.addError(objectToUpdate.getSecond());
        LOGGER.info(objectToUpdate.getSecond());
    }
} else {
    resultItem = createTaskResultItem(getMessage("TaskVerify_Success_Message"));
}
return Collections.singletonList(resultItem);
}

private TaskResultItem createTaskResultItem(String generalMessage) {
    return UpdateTaskResultBuilder.createTaskResultItem(generalMessage,
        ChangeLevel.OBJECT_TYPE, ChangeType.DATA_TRANSFORMATION);
}

private String getMessage(final String msgId) {
    return TasksMsgPacks.DataMigration.message(UpdateExecutedOrders52To60.class, msgId);
}

/**
 * ProductionCentre filters are not recommended
 *
 * @return objects to migrate
 */
private List<Pair<Long, String>> loadObjectsToMigrate() {
    ColumnDescriptor[] columnDescriptors = { new ColumnDescriptor(OBJECT_KEY),
                                              new ColumnDescriptor(OBJECT_NAME) };
    List<String[]> res = getSQLResult(columnDescriptors, SELECT_OBJECTS_IN_OLD_FORMAT);
    List<Pair<Long, String>> result = new ArrayList<Pair<Long, String>>();
    for (String[] row : res) {
        result.add(new Pair<Long, String>(Long.valueOf(row[0]), row[1]));
    }
    return null;
}

/**
 * @param columnDescriptors column descriptors
 * @param fromStatement SQL statement from and where clause
 * @return SQL result
 */
private List<String[]> getSQLResult(ColumnDescriptor[] columnDescriptors,
    String fromStatement) {
    FastLaneReader reader = new FastLaneReader();
    List<String[]> readerResult = reader.executeQueryUnformatted(columnDescriptors,
        false, fromStatement);

    if (readerResult != null) {
        return readerResult;
    } else {
        return Collections.EMPTY_LIST;
    }
}

/**
 * Execute update in a transaction
 *
 * @param objectToUpdate the object to update
 * @param taskResult write to stage XML

```

```

    * @return {@code true} if no error occurred
    */
    private void updateObjectInTransaction(final Pair objectToUpdate, final TaskResultItem
                                         taskResult) {
        try {
            TransactionInterceptor.callInTransactionImpl(new Callable<Void>() {
                @Override
                public Void call() throws DatasweepException {
                    updateObject(objectToUpdate, taskResult);
                    return null;
                }
            });
        } catch (Exception e) {
            LOGGER.error(getMessage("TaskUpdateTarget_Error_Message"), e);
        }
    }

    private void updateObject(final Pair objectToUpdate, final TaskResultItem taskResult) {
        // Implement update
    }

    /**
     * Clear task specific caches
     */
    protected void clearCaches() {
        PCContext.getServerImpl().getSiteCache().getATFactoryCache(PCContext.getFunctions()
            .getATDefinition(IMESChoiceList.ATDEFINITION_NAME).getKey()).clear();

        PCContext.getServerImpl().getSiteCache().getATFactoryCache(PCContext.getFunctions()
            .getATDefinition(IMESChoiceElement.ATDEFINITION_NAME).getKey()).clear();
        PCContext.getServerImpl().getSiteCache().getWorkCenterCache().clear();
    }
}

```

## Reference Documents

The following documents are available from the Rockwell Automation Download Site.

No.	Document Title	Part Number
A1	PharmaSuite Supported Platforms Guide	PSPG-RM084A-EN-E
A2	FactoryTalk ProductionCentre 10.4 Supported Platforms Guide	PRDCTR-RM104A-EN-E
A3	FactoryTalk ProductionCentre Release 10.4 Database Installation Guide	PRDCTR IN104A EN E
A4	PharmaSuite Technical Manual Administration	PSAD-RM008E-EN-E
A5	PharmaSuite Quality Certificate	10003305481/PUB
A6	FactoryTalk ProductionCentre Plant Operations Release 10.4 Server Installation Guide - JBoss Advanced	PCJBAD IN104A EN E
A7	PharmaSuite Technical Manual Installation - Enterprise Edition	PSEN-IN008E-EN-E
A8	PharmaSuite Technical Manual Configuration & Extension - Volume 4	PSCEV4-GR008E-EN-E
A9	PharmaSuite Technical Manual Installation - Building Block	PSBB-IN007E-EN-E
A10	PharmaSuite Building Blocks - Compatibility Matrix	PSBBCM-PA084A-EN-E
A11	PharmaSuite Technical Manual Configuration & Extension - Volume 1	PSCEV1-GR008E-EN-E

### TIP

To access the Rockwell Automation Download Site, you need to acquire a user account from Rockwell Automation Sales or Support.

The following documents are distributed with the PharmaSuite installation.

No.	Document Title / Section
C1	PharmaSuite Release Notes
C2	Java Documentation provided with the pre-configured Eclipse project

**TIP**

To access the Release Notes, use the following syntax:  
<http://<MES-PS-HOST>:8080/PharmaSuite/documentationandhelp/index.htm>, where  
 <MES-PS-HOST> is the name of your PharmaSuite server.

The following third-party documentation is available online as reference:

No.	Document Title / Web Site
D1	Unattended Installations (IzPack) <a href="http://docs.codehaus.org/display/IZPACK/Unattended+Installations">http://docs.codehaus.org/display/IZPACK/Unattended+Installations</a>

## Revision History

The following table describes the history of this document.

Changes related to the document:

Object	Description	Document
Screen captures	Wizard captures updated.	1.0

Changes related to "Introduction" (page 1):

Object	Description	Document
---	---	---

Changes related to "Upgrading an Installation" (page 3):

Object	Description	Document
Performing the Upgrade (page 6)	FactoryTalk ProductionCentre build and PharmaSuite versions updated.	1.0
Updating the Platform Updating FactoryTalk ProductionCentre (page 8)	FactoryTalk ProductionCentre version updated. Path to PharmaSuite-Help.ear updated.	1.0
Updating the Platform Prerequisites Checklist (page 7)	Step 7 deleted.	1.0
Updating the System Information Checklist (page 11)	Step 6: Java version updated to 1.8.0_144. ActiveMQ and the PharmaSuite upgrade engine run with the 64-bit version of Java.	1.0
Updating the System Prerequisites Checklist (page 13)	Step 1: FactoryTalk ProductionCentre version updated.	1.0
Migrating the Data Information Checklist (page 25)	Step 6: Java version updated to 1.8.0_144. ActiveMQ and the PharmaSuite upgrade engine run with the 64-bit version of Java.	1.0

Object	Description	Document
Migrating the Data Prerequisites Checklist (page 26)	Step 3: PharmaSuite version updated.	1.0
Migrating the Data Running the Data Migration Wizard (page 27)	Step 13: ActiveMQ version updated to 5.15.0.	1.0
Removing Duplicate Classes from the Classpath (page 37)	To retrieve the affected libraries, use the phase manager ( <b>mes_PhaseLibManager</b> form).	1.0

Changes related to "Technical Details of a System Update" (page 47):

Object	Description	Document
Updating ATRow Objects (page 59)	GHS statements added.	1.0
Updating FSM Objects (page 62)	Release-specific update task for FSMs added ( <b>UpdateFSMFrom83To84</b> ).	1.0

Changes related to "Technical Details of a Data Migration" (page 73):

Object	Description	Document
Migrating Filters of the Exception Dashboard to Support the Workflow Processing Type	Data migration task deleted.	1.0
Migrating Access Privileges to Retain the Previous Reason-specific Behavior	Data migration task deleted.	1.0
Migrating Barcodes of Equipment Entities	Data migration task deleted.	1.0
Migrating Barcodes of Stations	Data migration task deleted.	1.0
Migrating Transition Conditions of Activity Sets (page 75)	Data migration task added.	1.0
Migrating Usage List-related Tables (page 76)	Data migration task added.	1.0



Changes related to "Adapting the Upgrade Installer" (page [77](#)):

Object	Description	Document
---	---	---

- 
- 
- Rockwell Software PharmaSuite® - Technical Manual Installation - Upgrade
- 
-

**A**

- Adapt upgrade installer • 77
- Audience • 1
- Automated installation • 43
  - Console mode • 45
  - Wizard • 43

**B**

- Blockwise update • 73

**C**

- Checklist
  - Data migration • 25
  - FactoryTalk ProductionCentre update • 6
  - Update • 11
- Classpath (update) • 37
- Conventions (typographical) • 2
- Create upgrade task • 80

**D**

- Download Site • 91
- DSX conflict • 78

**F**

- FactoryTalk ProductionCentre update • 7

**G**

- Guidelines for implementing migration tasks • 85

**I**

- Implementing migration tasks • 85
- Intended Audience • 1

**L**

- Log level • 42
- log4j • 42
- Logging the upgrade • 42

**M**

- Merge DSX artifacts • 79
- Migrating the data • 25
- Migration • 73
  - Migrating transition conditions of activity sets • 75
  - Migrating usage list-related tables • 76
  - Task • 74

**R**

- Reference documents • 91
- Report of results • 38
- Resolve DSX conflict • 78
- Rockwell Automation Download Site • 91
- Rollout upgrade • 5

**S**

- Stage • 4

**U**

- Update • 47
  - Access Privilege objects • 61
  - Application objects • 68
  - AT definitions • 57
  - ATRow objects • 59
  - Checked-out objects • 51
  - DSX objects (generic DSX) • 53
  - EAR files • 70
  - FSM objects • 62
  - General strategy • 47
  - Image objects • 61
  - Library objects • 53
  - Message objects • 67
  - Report Design objects • 69
  - Report of customer modifications • 71
  - Task • 50
  - UDA definitions • 51
  - Versioning configuration objects • 68
- Updating

- 
- 
- Rockwell Software PharmaSuite® - Technical Manual Installation - Upgrade
- 
- 

Audit trail configuration • 9

Platform • 6

System • 11

Upgrading • 6