

webMethods Deployer User's Guide

Version 10.7

October 2020

This document applies to webMethods Deployer 10.7 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2004-2022 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <https://softwareag.com/licenses/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

Document ID: DEP-UG-107-20220711

Table of Contents

About This Guide	7
Document Conventions.....	8
Online Information and Support.....	8
Data Protection.....	9
1 Introduction to webMethods Deployer	11
About webMethods Deployer.....	12
Starting Deployer.....	12
Deployer Interfaces.....	13
Concurrent and Sequential Deployment.....	13
Logging.....	14
2 Getting Started with Deployer	17
Runtime-Based Deployment Quick Start.....	18
Repository-Based Deployment Quick Start.....	19
3 Deploying Assets from Source webMethods Runtimes	21
About Runtime-Based Deployment.....	22
Connecting to webMethods Source and Target Runtimes.....	22
Creating Runtime Deployment Projects.....	31
Creating a Deployment Set from Runtime Assets.....	32
Identifying Source Servers for a Deployment Set.....	33
Adding Assets to Broker, ProcessModel, My webMethods Server, or Optimize Deployment Sets.....	34
Adding Assets to an IS & TN Deployment Set.....	35
Deploying ACLs through a Deployment Set.....	44
Resolving Dependencies.....	45
Using Deletion Sets.....	47
Creating a Build.....	51
Rebuilding a Build.....	53
Exporting and Importing a Build.....	53
4 Deploying Assets from a Source Asset Repository	55
About Repository-Based Deployment.....	56
Building Composites.....	57
Connecting to a Source Repository.....	69
Connecting to Target webMethods Servers.....	70
Creating Repository Deployment Projects.....	87
Creating a Deployment Set from Repository Assets.....	88
Identifying the Source Repository for a Deployment Set.....	88
Selecting Composites.....	89
Selecting Individual Assets from Composites.....	89
Deploying ACLs through a Deployment Set.....	90

Resolving Dependencies.....	91
Resolving Conflicts.....	94
Using Deletion Sets.....	95
5 Product Build Properties and Supported Assets for Repository-Based Deployment.....	99
About Product Build Properties and Supported Assets.....	101
ActiveTransfer.....	101
AgileApps.....	106
API Gateway.....	107
Application Platform.....	115
BPM Process Development.....	117
Broker.....	122
Business Rules.....	123
Digital Event Services Assets.....	124
Event Routing.....	125
Event Server.....	125
Integration Server.....	126
Mobile Support.....	189
My webMethods Server.....	190
Optimize.....	194
Trading Networks.....	195
Task Engine.....	198
Universal Messaging.....	198
Other Product Assets.....	200
6 Managing Deployment Projects.....	201
About Configuring Project Settings.....	202
Settings for All Projects.....	202
Settings for Runtime-Based Deployment Projects.....	202
Settings for Repository Based Deployment Projects.....	207
Creating a Project.....	209
Exporting and Importing Project Properties.....	212
Managing User Permissions for Project Tasks.....	213
Adding and Viewing Instructions or Notes About a Project.....	215
Update the Settings for a Project.....	215
Deleting a Project.....	216
7 Mapping a Deployment Project to Target Servers.....	217
About Mapping a Project.....	218
Creating Target Groups.....	219
Mapping a Project to Target Servers and Target Groups.....	225
Mapping a Project with a Deletion Set for Runtime-based Deployment.....	227
Troubleshooting the Deployment Map.....	227
Exporting and Importing a Map.....	228
Substituting Configuration Values.....	229
Exporting and Importing Substitute Configuration Values.....	231
8 Deploying a Project.....	233

Generating a Checkpoint.....	234
Deploying a Project.....	234
Post-Deployment Tasks.....	237
Rolling Back Target Servers.....	238
9 Using Deployer Commands.....	241
Installing Command Line Interface Only.....	242
Creating and Running Scripts.....	242
Specifying Logon Parameters.....	245
Error Handling and Logging.....	248
General and Project Commands.....	248
Build Commands.....	252
Commands for Repository-Based Deployment.....	257
Map Commands.....	257
Deployment Commands.....	261
10 Using Project Automator.....	267
Exporting Projects to Project Automator.....	268
Using Handles Instead of Passwords.....	269
Error Handling and Logging for Project Automator.....	271
Root Tag.....	271
Identifying Deployer.....	272
Setting Up Aliases for Source and Target Servers.....	273
Creating Projects.....	308
Running Project Automator.....	324
11 Assets with a Specific Deployment Setup.....	325
Deploying Process Models with E-Forms.....	326
Deploying Optimize Assets.....	327

About This Guide

- Document Conventions 8
- Online Information and Support 8
- Data Protection 9

This document explains how to use webMethods Deployer to deploy assets from source webMethods servers or development environments to target webMethods servers.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Introduction to webMethods Deployer

■ About webMethods Deployer	12
■ Starting Deployer	12
■ Deployer Interfaces	13
■ Concurrent and Sequential Deployment	13
■ Logging	14

About webMethods Deployer

webMethods Deployer is a tool you use to deploy user-created assets that reside on source webMethods runtimes or repositories to target webMethods run-time components (*runtimes*). For example, you might want to deploy assets you have developed on servers in a development environment (the source) to servers in a test or production environment (the target).

With Deployer, you can deploy *user-created* assets such as Integration Server packages. However, you cannot deploy webMethods components that have been installed by the Software AG Installer as part of a product. If you want such components to reside on target runtimes, you must install them using the Software AG Installer. However, you can partially deploy system (Wm*) packages for some webMethods adapters, such as the adapter packages for WmAS400, WmSAP, WmJDBCAdapter, and WmMQAdapter. In partial deployment of system packages, configurations or artifacts that reside in the system packages on the source server can be deployed to the target server. To determine if system packages can be partially deployed for a particular product, refer to the product documentation.

Deployer is installed on Integration Server. This help can refer to Integration Server as the host server for Deployer, and as a runtime from which you can deploy assets using Deployer. You can also use Microservices Runtime as the host server, and as a runtime from which you can deploy assets.

Deployer supports two scenarios for deploying assets:

- In a *runtime-based deployment* scenario, Deployer deploys assets from webMethods runtimes to which Deployer is connected.

Deprecation Notice: Beginning with release version 10.5, runtime-based deployment is deprecated. Software AG recommends that you use the Asset Build Environment and repository-based deployment instead.

- In a *repository-based deployment* scenario, Deployer deploys assets built from sources in a development environment or version control system (VCS) and stored in a repository.

Starting Deployer

Deployer starts automatically when you start its host Integration Server.

To open the Deployer graphic user interface, enter this URL in an Internet browser, where *IS_host:IS_port* is the host name and port of the Integration Server that hosts Deployer:

```
http://IS_host:IS_port/WmDeployer
```

Deployer and Integration Server use the same log on user name and password. If you just installed Deployer with a new Integration Server, the defaults are user name Administrator and password manage.

Deployer Interfaces

Deployer offers a graphical user interface (GUI), a command line interface (CLI), and a Project Automator tool. You can create scripts that execute the CLI commands automatically. Use the Project Automator to automate creating projects by specifying project information in an XML file.

The following table shows which tasks you can perform from each type of Deployer interface.

Task	GUI	Command Line	Project Automator
Add and view instructions or notes about projects.	X		
Authorize groups to work on projects.	X		
Configure communication between Deployer and the source servers or repositories and target servers.	X		X
Create project builds.	X	X	X
Create projects.	X		X
Define deployment and deletion sets.	X		X
Delete projects and set default properties for projects	X	X	
Export and import project properties and deletion set definitions.	X	X	
Generate a checkpoint or roll back target servers, list and display rollback reports.	X	X	
Generate progress reports.	X		
List, create, display, or delete deployment candidates; simulate deployments; deploy; list and display simulation and deployment reports.	X	X	
List, display, export, import, and delete deployment maps.	X	X	
List, export, and import builds; display build contents; list and display build reports.	X	X	
Map deployment and deletion sets to target servers.	X		X
Select assets from an ACDL descriptor file	X	X	X

Concurrent and Sequential Deployment

Deployer supports both sequential and concurrent asset deployment.

In a *concurrent* deployment, Deployer uses the host Integration Server thread pool to create a separate thread in order to deploy the assets to all target servers simultaneously. This is the default setting. You should configure the minimum and maximum threads in the pool to accommodate your system. For more information about setting the minimum and maximum thread pools, see *webMethods Integration Server Administrator's Guide*. Deployer writes records to the Audit Log to indicate that the main deployment thread has created new deployment threads for the specific target server. This allows you to track which thread belongs to each deployment request.

In a *sequential* deployment, Deployer deploys the assets to all target servers one by one in the order that the targets were added to the deployment map. If the project is not set to use concurrent deployment, Deployer deploys assets sequentially.

You can set concurrent deployment for all projects in the system when you set the default settings for Deployer as described in [“Settings for All Projects” on page 202](#). If the default setting is set to sequential deployment and you want to use concurrent deployment for a specific project (or vice versa), you can override the default setting for that project during creation. For more information about overriding the default settings for a project, see [“Creating a Project” on page 209](#).

Logging

Deployer writes audit log entries to these logs:

- The Deployer GUI audit log. This log contains information about actions that users perform using the Deployer GUI, such as creating builds and deploying.
- The Deployer command line audit log. This log contains error messages written by Deployer commands executed by users.
- The Deployer Project Automator audit log. This log contains information and error messages written by the Project Automator during execution of an XML file.

Deployer writes journal entries to the Integration Server server log. The server log contains information about operations and errors that occur on Integration Server, such as the starting of Integration Server subsystems and the loading of Integration Server packages. For complete information about the Integration Server server log, see *webMethods Integration Server Administrator's Guide*.

Enabling or Disabling Audit Logging for the Deployer User Interface

In Deployer, go to **Settings > Audit Logging Settings** and specify whether to enable or disable audit logging for user actions taken through the Deployer user interface.

To view the audit log, from the Integration Server Administrator go to the **Logs > Audit** page. The following table describes the columns in the audit log.

Column	Description
Time Stamp	Date and time the entry was written to the log.

Column	Description						
Request Type	Type of action Deployer performed (for example, Create , Build , or Deploy).						
Message	Message that describes the action.						
Status	Outcome of the action (for example, Success or Failed).						
User Id	User name under which Deployer performed the action.						
Server	Details about the server. See Type , Alias , and Host IP:Port .						
	<table border="1"> <tr> <td>Type</td> <td>Type of server on which the action was performed; can include the Integration Server that hosts Deployer and source and target servers.</td> </tr> <tr> <td>Alias</td> <td>Name assigned to the server in Deployer.</td> </tr> <tr> <td>Host IP:Port</td> <td>IP address and port for the server on which the action was performed.</td> </tr> </table>	Type	Type of server on which the action was performed; can include the Integration Server that hosts Deployer and source and target servers.	Alias	Name assigned to the server in Deployer.	Host IP:Port	IP address and port for the server on which the action was performed.
Type	Type of server on which the action was performed; can include the Integration Server that hosts Deployer and source and target servers.						
Alias	Name assigned to the server in Deployer.						
Host IP:Port	IP address and port for the server on which the action was performed.						
Thread ID	Unique identification for each Deployer action. If an action fails, the thread ID helps you identify the failed action through the Audit Log page in Deployer. Using this, the user can identify the failed activity.						

To change this display, use the **Log display controls** area at the top of the page and then click **Refresh**. The changes remain until you change them again, or until you shut down the Integration Server that hosts Deployer, whichever comes first.

2 Getting Started with Deployer

- Runtime-Based Deployment Quick Start 18
- Repository-Based Deployment Quick Start 19

Runtime-Based Deployment Quick Start

The runtime-based deployment process involves the following basic stages and in each stage you perform the general tasks listed for that stage:

Stage 1: Define a deployment project

You use Deployer to create connections to source and target webMethods runtimes. You then define a deployment set, for which you select user-created assets to include in the project directly from source webMethods runtimes.

Task	See...
Start Deployer.	“Starting Deployer” on page 12
Set up connections to source and target servers.	<ul style="list-style-type: none">■ “Connecting to Integration Server and Trading Networks” on page 23■ “Connecting to Optimize Servers” on page 30■ “Connecting to My webMethods Server” on page 25■ “Connecting to webMethods Broker” on page 28■ “Connecting to BPM Process Model Servers” on page 26■ “Creating Target Groups” on page 219
Create a project.	“Creating a Project” on page 209
Define deployment and deletion sets.	<ul style="list-style-type: none">■ “Creating a Deployment Set from Runtime Assets” on page 32■ “Creating a Deletion Set” on page 47
Add assets to the deployment set.	“Adding Assets to a Runtime-Based Deletion Set” on page 49

Stage 2: Build the project, map and deploy assets to runtimes

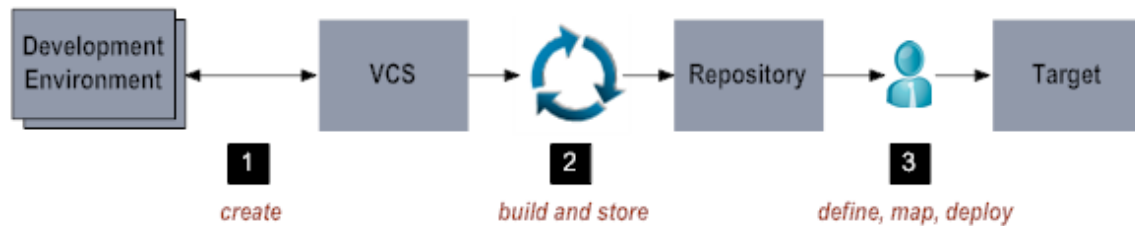
You build your project for deployment to the target runtimes. When you build a project, Deployer creates a file that contains the actual assets referenced in the project. If you later change the project, or if the build contains assets that you know have changed on the source runtimes, you can re-create the build to bring it up to date. Map the contents of the project build to target runtimes and deploy the assets.

Task	See...
Build the project.	“Creating a Build” on page 51

Task	See...
Map assets to target servers and target groups.	“Mapping a Project to Target Servers and Target Groups” on page 225
Deploy the project.	“Deploying a Project” on page 234

Repository-Based Deployment Quick Start

The following graphic illustrates the asset deployment process in repository-based deployment:



The repository-based deployment process involves the following basic stages and in each stage you perform the general tasks listed for that stage:

Stage 1: Create assets on your development environment.

You create assets in the development environment and save them on a server, or check them into a version control system (VCS).

Stage 2: Build the composites and descriptors

You use the master build script to build the composite and descriptor files from the assets, and store these files in the repository. For the tasks that you do in this stage, see [“Building Composites” on page 57](#).

Stage 3: Define a deployment project, map and deploy the assets to target servers

You define a *deployment project* and *deployment sets*. You select user-created assets from the ACDL composite stored in the repository to identify the assets to include in the project.

Task	See...
Start Deployer.	“Starting Deployer” on page 12
Set up a connection to the source repository.	“Connecting to a Source Repository” on page 69
Set up connections to the target servers. ■	“Connecting to a Target ActiveTransfer Server” on page 70

Task	See...
	<ul style="list-style-type: none">■ “Connecting to a Target API Gateway Server” on page 71■ “Connecting to a Deployment Endpoint for Digital Event Services” on page 72■ “Connecting to Integration Server and Trading Networks” on page 23■ “Connecting to Optimize Servers” on page 30■ “Connecting to a Target Application Platform Server” on page 75■ “Connecting to My webMethods Server” on page 25■ “Connecting to a Target AgileApps Cloud Server” on page 82■ “Connecting to a Target Event Server” on page 77■ “Connecting to webMethods Broker” on page 28■ “Connecting to BPM Process Model Servers” on page 26■ “Connecting to a Target Business Rules Integration Server” on page 85
Create a project.	“Creating a Project” on page 209
Define a deployment and deletion sets.	<ul style="list-style-type: none">■ “Creating a Deployment Set from Repository Assets” on page 88■ “Creating a Deletion Set” on page 95
Add assets to the deployment set.	“Creating a Deployment Set from Repository Assets” on page 88

After defining the deployment set, you map the contents of the deployment project to target servers and deploy the assets.

Task	See...
Map assets to target servers or target groups.	“Mapping a Project to Target Servers and Target Groups” on page 225
Deploy the project.	“Deploying a Project” on page 234

3 Deploying Assets from Source webMethods Runtimes

- About Runtime-Based Deployment 22
- Connecting to webMethods Source and Target Runtimes 22
- Creating Runtime Deployment Projects 31
- Creating a Deployment Set from Runtime Assets 32
- Identifying Source Servers for a Deployment Set 33
- Adding Assets to Broker, ProcessModel, My webMethods Server, or Optimize Deployment Sets 34
- Adding Assets to an IS & TN Deployment Set 35
- Deploying ACLs through a Deployment Set 44
- Resolving Dependencies 45
- Using Deletion Sets 47
- Creating a Build 51
- Rebuilding a Build 53
- Exporting and Importing a Build 53

About Runtime-Based Deployment

Deprecation Notice: Beginning with release version 10.5, runtime-based deployment is deprecated. Software AG recommends that you use the Asset Build Environment and repository-based deployment instead.

In runtime-based deployment, you deploy assets directly from webMethods source runtimes to target runtimes.

The following webMethods runtimes support runtime-based deployment:

- webMethods Brokers.
- Integration Servers, including hosts for Trading Networks Servers.
- My webMethods Server.
- Process Engine (BPM) runtimes (an Integration Server that hosts the webMethods Process Engine and executes business processes)
- Optimize runtimes (an Integration Server that hosts the Optimize Analytic Engine).

You can deploy *user-created* assets using Deployer. You cannot deploy system components that were installed by the Software AG Installer as part of a product installation. For example, you can deploy Integration Server packages that were created by users, but you cannot deploy Integration Server system packages that were installed, such as the WmPRT package (Process Engine). If you want such components on target servers, you must install them using the Software AG Installer. However, you can partially deploy system (Wm*) packages for some webMethods adapters, such as the adapter packages for WmAS400, WmSAP, WmJDBCAdapter, and WmMQAdapter. In partial deployment of system packages, configurations or artifacts that reside in the system packages on the source server can be deployed to the target server. To determine if system packages can be partially deployed for a particular product, refer to the product documentation.

Connecting to webMethods Source and Target Runtimes

When deploying assets from source runtimes, the *source* servers are those servers from which you select assets for deployment. The *target* servers are the target runtimes to which Deployer deploys assets from the source server.

For every source and target server you add, you select the version of the webMethods runtime for that server. For example, if you are adding a target Integration Server that is running version 9.12, you would select 9.12 for the version. The version you select affects the source and target servers you can include in the project. After you select the source servers, Deployer displays as available for mapping only target servers that have compatible versions with the source servers. For example, if you select source servers of version 9.12, Deployer displays for mapping only target servers and target groups of version 9.12.

The following table lists all version compatibilities:

Source Version	Compatible Target Versions
7.1	7.1
8.0	8.0, 8.2
	8.2 targets support only the following runtimes from 8.0 sources:
	<ul style="list-style-type: none"> ■ Integration Server ■ Trading Networks ■ Process Engine (BPM process models)
8.2	8.2
9.0	9.0
9.5	9.5
9.6	9.6
9.7	9.7
9.8	9.8
9.9	9.9
9.10	9.10
9.12	9.12
10.0	10.0
10.1	10.1
10.2	10.2
10.3	10.3
10.4	10.4

You can import builds from projects created with Deployer version 8.2 or lower to Deployer 9.0 or higher for deployment. For example, you could import a build from a version 8.0 project into Deployer 9.0.

Connecting to Integration Server and Trading Networks

All connections you create for Integration Servers and Trading Networks servers are *remote server aliases*. A remote server alias contains the connection information required to connect to a remote Integration Server or Trading Networks server. For more information about remote servers, and instructions on defining them, see *webMethods Integration Server Administrator's Guide*.

➤ **To connect to source and target Integration Servers and Trading Networks servers**

1. In Deployer, go to the **Servers > IS & TN** page.

2. Click **Add Remote Server Alias**.

Deployer opens the Integration Server Administrator to the **Settings > Remote Servers > Create Alias** page of the Integration Server that hosts Deployer.

3. In Integration Server Administrator, define the remote servers by completing the fields in the **Remote Server Alias Properties** area as described in *webMethods Integration Server Administrator's Guide*. You should define the following as remote servers:

- All source Integration Servers and Trading Networks servers.
- All target Integration Servers and Trading Networks servers.
- The Integration Server that hosts Deployer, if you will be using it as a source or target server (that is, define the Integration Server as a remote server to itself).

4. In Deployer, go to the **Servers > IS & TN** and click **Refresh this Page**.

Deployer displays the new server alias to the **Remote Servers List** area of the **Servers > IS & TN** page.

5. Select the version of the Integration Server hosting the source or target runtime from the **Version** box.

For example, if the host Integration Server is running version 9.12, you would select **9.12**. For information about selecting the version, see [“Connecting to webMethods Source and Target Runtimes” on page 22](#).

6. Install the WmDeployerResource package on each Integration Server.

The WmDeployerResource package is the implementation of the operation endpoints for Integration Server and Trading Networks. Install the package as follows:

a. In Deployer, go to the **Servers > IS & TN** page.

The page lists all Integration Servers you defined as remote servers.

b. In the **Install** column, select the check box next to each Integration Server on which you want to install the WmDeployerResource package.

c. Click **Install**.

7. Click **Save**.

Connecting to My webMethods Server

Use the following procedure to set up connections to source and target My webMethods Servers.

> To connect to source and target My webMethods Servers

1. In Deployer, go to the **Servers > MWS** page.
2. For every source and target My webMethods Server, click **Configure MWS Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the My webMethods Server. For information about selecting the version, see “Connecting to webMethods Source and Target Runtimes” on page 22.
Root folder aliases	My webMethods Server aliases to use as root folders when selecting pages to deploy. Separate the folders using commas.
Include security dependencies	Whether to include the following in the dependencies list for My webMethods Server assets when creating an MWS deployment set: <ul style="list-style-type: none"> ■ Security realms that contain the assets. ■ User/group/role references in the assets' security ACLs. <p>If the dependencies do not exist on the target My webMethods Servers, include them in the list. You will have to include them in the deployment set. Exclude the dependencies if they do exist on the target My webMethods Servers. For information about resolving dependencies, see “Resolving Dependencies” on page 45.</p>
Maximum Folder Object Count	Maximum number of assets to display within My webMethods Server folders when you are defining and choosing assets to include in an MWS deployment set.

Field	Entry
Maximum Folder Depth	Maximum number of My webMethods Server folder levels to display when you are defining and choosing assets to include in an MWS deployment set.
Enable additional MWS logging	Whether to log debug information about selected assets to source My webMethods Server logs, and assets that Deployer deploys to target My webMethods Server logs.
Exclude Core Task Engine Dependencies	Whether to exclude Task Engine portlets from the dependencies list for task application assets. Exclude the portlets from the list if the target My webMethods Servers host the Task Engine; the portlets are installed with the Task Engine. Include the portlets if the target My webMethods Servers do not host the Task Engine; you will have to include the portlets in the deployment set. For information about dependencies, see “Resolving Dependencies” on page 45 .
Cache Timeout	Length of time queries should remain in the cache unless the cache capacity is exceeded.
Use SSL	Whether Deployer should use SSL to connect to the My webMethods Server.
	<p>Note: You can only use SSL if the My webMethods Server is configured to use SSL. Configure the My webMethods Server's HTTPS port to <i>not</i> request client certificates. For instructions on defining the HTTPS port, see <i>Administering My webMethods Server</i>.</p>

3. Click **Configure**. To test the connection, click .


Connecting to BPM Process Model Servers

A BPM process model server is an Integration Server that hosts the webMethods Process Engine and executes business processes.

➤ To connect to source and target BPM (ProcessModel) servers


1. Make sure the Process Audit Log database component is installed and Integration Server is configured to write to it. For instructions, see *Installing Software AG Products*.
2. In Deployer, go to the **Servers > BPM(ProcessModel)** page.
3. For every source or target ProcessModel server, click **Configure BPM(ProcessModel) Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the server. For information about selecting the version, see “Connecting to webMethods Source and Target Runtimes” on page 22.
Use SSL	Whether Deployer should use SSL to connect to the server.

4. Click **Configure**. To test the connection, click .
5. In Designer, duplicate the logical-to-physical server mapping (defined for each ProcessModel server) on the source and target ProcessModel servers for each model you want to deploy. In the Integration Server Administrator for each of the servers, do the following:
 - a. Define the physical servers in the mapping as remote servers. For instructions, see *webMethods Integration Server Administrator's Guide*.


Note:

Each Integration Server hosting a process model in a cluster must be configured to use the same alias name and port number as the remote alias defined for the cluster. For more information about deploying to clustered Integration Servers, see [“Deploying to Clustered Integration Servers” on page 223.](#)

- b. Go to the **Packages > Management** page and click  for the WmDesigner package.
- c. Click **Add Logical Server** and complete these fields:

Field	Entry
Name	Name of a logical server in the mapping for the ProcessModel server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
	Note:

Field	Entry
	In clusters, both the source and target ProcessModel servers must use the same logical server name.
Physical Server	Physical server to which the logical server is mapped. This should be the same value as the remote server alias name you defined in step 5a.

- d. Click **Add Logical Server**.
 - e. Repeat these steps to duplicate the rest of the mapping.
 - f. Repeat these steps for every process model you want to deploy.
6. Install the WmDeployerResource package on each ProcessModel server that will run process steps. In Deployer, go to the **Servers > IS & TN** page; the page lists all ProcessModel servers you defined as remote servers. In the **Install** column, select the check box next to each ProcessModel server and click **Install**.
 7. If a process model to deploy includes a task, go to the **Packages > Management** page on the model's source and target ProcessModel servers, click  for the WmTaskClient package, and identify the My webMethods Server that hosts the task.

Connecting to webMethods Broker

Important:

webMethods Broker has been deprecated.

For each source and target Broker, Deployer must connect to the Broker Server that controls that Broker.

> To connect to source and target Broker Servers

1. In Deployer, go to the **Servers > Broker** page.
2. For every source or target Broker, click **Configure Broker Server** and complete these fields:

Field	Entry
Name	Name to assign to the Broker Server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the Broker Server.

Field	Entry
Port	Port for the Broker Server.
Version	Version of the Broker Server that matches the version of the project as defined by the host server alias. For information about selecting the version, see “Connecting to webMethods Source and Target Runtimes” on page 22.
Broker Name	Name of the source or target webMethods Broker.
Client Group	Client group for Deployer to use to access the source or target Broker Server. For target Broker Servers, type <code>admin</code> . <p>Note: To connect, Deployer must belong to the specified Broker client group and have access permission.</p>
Context	JNDI context to use when the Broker Server serves as a JNDI provider.
Client Authentication	Whether Deployer should use client authentication to connect to the Broker Server. Select: <ul style="list-style-type: none"> ■ None to connect to the Broker Server without any client authentication. ■ SSL to connect to the Broker Server using SSL authentication. If you select this option, you must complete the Deployer Keystore, Keystore Type, Keystore Password, DeployerTruststore, and Truststore Type fields. <p>Note: You can only use SSL if the Broker Server is configured to use SSL authentication.</p> ■ Basic Authentication to connect to the Broker Server using basic authentication. If you select this option, you must complete the Username and Password fields. <p>Note: You can only use Basic Authentication if the Broker Server is configured to use basic authentication.</p>
Deployer Keystore	Full path to Deployer's keystore file. The keystore contains the SSL credentials (private key and signed certificate) that the Broker Server uses to authenticate Deployer's identity and establish an SSL connection. Required if you specify SSL for Client Authentication .
Keystore Type	File type of Deployer's keystore file. Required if you specify SSL for Client Authentication .
Keystore Password	Password that Deployer uses to access its keystore file. Required if you specify SSL for Client Authentication .

Field	Entry
Deployer Truststore	Full path to Deployer's truststore file. The truststore contains the trusted roots for Deployer's SSL certificates. Required if you specify SSL for Client Authentication .
Truststore Type	File type of Deployer's truststore file. Required if you specify SSL for Client Authentication .
Username	Basic authentication user name. Required for Basic Authentication .
Password	Basic authentication password. Required for Basic Authentication .

3. Click **Configure**. To test the connection, click .

Connecting to Optimize Servers

Use the following procedure to set up connections to Optimize servers.

> To connect to source and target Optimize servers

1. In Deployer, go to the **Servers >Optimize** page.
2. For every source and target Optimize server, click **Configure Optimize Server**. In the **Configure Optimize Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server. Note: You cannot use "localhost" for Optimize servers.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Optimize server. For information about selecting the version, see "Connecting to webMethods Source and Target Runtimes" on page 22.
Use SSL	Whether Deployer should use SSL to connect to the server.

3. Click **Configure**. To test the connection, click .

Updating Properties for Source and Target Servers

Use the following procedure to update the properties for webMethods source and target servers.

Note:

For Integration Server and Trading Networks remote server aliases, you can update only the **Version** property using Deployer. You can update all other properties for Integration Server and Trading Networks remote server aliases using Integration Server Administrator. For more information about updating remote server aliases, see *webMethods Integration Server Administrator's Guide*. For information about refreshing Integration Server and Trading Networks remote server aliases in Deployer, see step 4 in [“Connecting to Integration Server and Trading Networks” on page 23](#).

➤ Editing source and target server properties

1. In Deployer, go to the **Servers > server** page.
2. To update the version of the server, select the version of the server hosting the source or target from the **Version** box and click **Save**. You can update the version for more than one server at a time.

For information about selecting the version of the server, see [“Connecting to webMethods Source and Target Runtimes” on page 22](#).

3. To update additional server properties, click the name of the server to edit.

Deployer displays the server properties in the right-hand pane.

4. Update the details for the server as necessary.
5. Click **Save**.

Creating Runtime Deployment Projects

A *runtime deployment project* identifies the user-created assets on source servers that you want to deploy to target servers. To create a project, you assign the project a name and set its properties, and then you authorize users to perform the project tasks.



When you create a project, Deployer automatically creates an HTML home page for the project. You can modify this page to contain instructions or notes about the project that you want users to view. For example, you might want to list the target servers for the users who will perform the mapping task, or you might want to provide instructions for users who will test the deployed solution.

For instructions about creating and managing deployment projects, see [Managing Deployment Projects](#).

Creating a Deployment Set from Runtime Assets

You identify the assets to include in the project using *deployment sets*. Each deployment set identifies the user-created assets you want to deploy from one type of source server to a target server. A single project can include deployment sets with assets from different types of webMethods applications. For example, a single project can include Integration Server deployment sets and ProcessModel deployment sets.

> To create a deployment set

1. In Deployer, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. In the right-hand pane, click  **Define**.
5. Click **Create Set**.
6. If you are creating a deployment set for runtime-based deployment, perform the following:
 - a. From the **Type** box, select the type of deployment set you want to create.
 - b. From the **Set** box, select **Deployment**.
7. Specify the following fields:

Box	Entry
Name	Name to use for the deployment set. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Description	Description for the deployment set. The description length has no limit and can include any characters.
Maximum TN Assets to Display	Number of Trading Networks assets Deployer should display. Depending on your browser, Deployer might not be able to display more than 1000 assets for Trading Networks. If the source server hosts more than 1000 assets, use this field with the with the All other assets field to reduce the number of assets displayed.

Box	Entry
Packages (IS & TN deployment set)	<p>After you choose the source servers, Deployer will display all packages on the servers. You can use this field to narrow the display. Type a regular expression that specifies the text that the package names must contain in order to be displayed.</p> <p>The following are examples of regular expressions:</p> <ul style="list-style-type: none"> ■ To narrow display to packages whose name starts with <i>string</i>, use the following: <i>string.*</i> ■ To narrow display to packages whose name ends with <i>string</i>, use the following: <i>.*string\$</i>
All other assets	<p>After you choose the source servers, Deployer will display all assets on those servers. You can use this field to narrow the display. Specify a regular expression that specifies the text that the asset names must contain in order to be displayed. For examples, see the Packages field.</p>

8. Click **Create**.

Identifying Source Servers for a Deployment Set

You cannot define the same server alias as both a source and target server in a deployment set. If a server you want to use as a source is not in the list of available servers, you have not yet created the alias for the server. Connect to the server as described in [“Connecting to webMethods Source and Target Runtimes” on page 22](#) and then click **Refresh this Page** to update the list of servers. Also check [“About Runtime-Based Deployment” on page 22](#) for the list of webMethods runtimes that support deploying from source runtimes.

➤ To identify source servers for runtime-based deployment

1. On the **Deployer > Projects > project > Define** page, in the left-hand pane in the **Name** column, click the deployment set for which to identify source servers.
2. In the **Select** column of the **Select Source Servers** area, select the check box next to each source server that contains assets to add to the deployment set.

Deployer lists the version of each source server in the **Version** column. You must select source servers with the same version for the deployment set. You cannot include source servers with different versions in a deployment set. For information about selecting the versions of source servers, see [“Connecting to webMethods Source and Target Runtimes” on page 22](#).

Note:

For a BPM (ProcessModel) deployment set, you can select only one source server. If you want to deploy process models from more than one ProcessModel server, you must define a deployment set for each ProcessModel server.

3. Click **Save**.

Deployer refreshes the **Select Source Servers** list to display only source servers that share the same version as the selected source servers. To display all of the available source servers, deselect the source servers and click **Save**.



Adding Assets to Broker, ProcessModel, My webMethods Server, or Optimize Deployment Sets

Based on which product assets go into the deployment set, you should consider the following before adding assets to the set:

- Deployer cannot display more than 2500 assets at once, so it might not display all assets. Software AG recommends displaying smaller sets of assets and creating smaller deployment sets. When creating a deployment set with My webMethods Server assets, the **Root folder aliases** field in the MWS Server configuration (see [“Connecting to My webMethods Server” on page 25](#)) controls the Page assets that are displayed for each My webMethods Server. By default, this field is set to folder.public. If your Public Folders and its subfolders contain hundreds of pages, displaying these assets can take a long time. To deploy smaller sets of assets, change the **Root folder aliases** field to specify a folder that is deeper within the Public Folders hierarchy and create a deployment set for those assets.
- When deploying BPM process models, Deployer copies the generation receipt from the source environment to the target environment. Because the logical-to-physical server mapping might contain old data, the regeneration process connects to the previous logical server and cleans up the process-related assets there (for example, deleting associated triggers and services), in addition to creating the new assets on the new logical server. With no generation receipt, this mapping information is not available and the clean-up procedure cannot occur. If you deploy multiple versions of the same process model in a single deployment set and configure the deployment set to enable both processes on the target, the version that is deployed to the target last is the one that is enabled. Use webMethods Monitor to ensure that the proper version is enabled in the target environment.

➤ To add assets to Broker, ProcessModel, MWS, or Optimize deployment sets

1. In the **Deployment Sets** area, under the deployment set to which to add assets, click the **Broker, ProcessModel, MWS, or Optimize** folder. In the right-hand pane, Deployer lists the source servers of the type you selected.
2. In the right-hand pane, open the tree to show the assets on the source servers, then select the check box next to each asset to add to the deployment set. Keep in mind the following:

For...	Note
ProcessModel	The process models displayed are those that were "Built for execution" on the Integration Server.
MWS	The My webMethods Server folder is listed twice within its directory, as a container preceded by  and as an asset preceded by  . If you want to add a folder with all the assets it contains to the deployment set, select the folder where it appears next to the square icon. If you want to add individual assets in the folder without adding the folder itself, open the folder where it appears as a container and click the assets to add.

3. Click **Save**. Deployer shows the selected assets in the left-hand pane under the **Broker**, **ProcessModel**, **MWS**, **Optimize**, or **Business Rules** folder for the deployment set.

Removing Process Models from a Deployment Set

When you add a process model to a ProcessModel deployment set and then add referenced assets that reside on Integration Servers, Deployer shows the referenced assets as children of the process model. If you want to remove a process model from a deployment set, clear the check box next to the process model under the tree. This removes the process model from the deployment set; however, the dependencies must be removed manually.

Adding Assets to an IS & TN Deployment Set

You can add the following types of user-created assets to an IS & TN deployment set:

- Integration Server administrative assets such as ports, users, groups, and scheduled tasks, packages, and web service descriptors.
- Integration Server packages.
- webMethods files.
- Trading Networks assets.

Your source and target Integration Servers might have assets in common, and you do not need to deploy them from one environment to another. You can improve Deployer performance by excluding these assets from deployment sets. For instructions, see [“Excluding Common Assets” on page 44](#).

Integration Server introduces new assets from time to time and not all the assets can be deployed using runtime-based deployment. In a runtime-based deployment, Deployer may not be able to recognize new types of Integration Server assets. If a package contains such assets, Deployer displays those assets as unknown types and shows them as unresolved dependencies. While deploying, select **Ignore** to bypass dependency checking. Those assets cannot be selected individually for deployment. It is recommended to use repository-based deployment.

Adding Integration Server Administration Assets

Use the following procedure to add Integration Server administration assets.

> To add Integration Server administration assets

1. In the **Deployment Sets** area, under the deployment set to which to add Integration Server administration assets, click the **Administration** folder. In the right-hand pane, Deployer lists the source Integration Servers you identified.
2. In the right-hand pane, open the tree to show the administration assets on the source Integration Servers, select the check box next to each asset to add to the deployment set, and then click **Save**. Deployer shows the selected assets in the left-hand pane under the **Administration** folder for the deployment set.
3. If you are not going to add any more assets to the deployment set, go to [“Resolving Dependencies” on page 45](#).
4. If you added JMS triggers to the deployment set, create the same JMS connection aliases on the target Integration Servers that exist on the source Integration Servers. For instructions, see *Using webMethods Integration Server to Build a Client for JMS*.

Note:

If the JMS connection aliases on the target Integration Servers do not have the same names as on the source Integration Servers, the JMS triggers will not be enabled after deployment.

Adding an Entire Package

> To add an entire package

1. In the **Deployment Sets** area, under the deployment set to which to add packages, click the **Packages** folder. In the right-hand pane, Deployer lists all source Integration Servers.
2. In the right-hand pane, open the tree to show the packages on the source Integration Servers, select the check boxes next to the packages to add in their entirety, and then click **Save**. Deployer shows the entire package icon (📦) for the selected packages in the left-hand pane under the **Packages** folder and in the right-hand pane, and a black check mark for the packages in the right-hand pane.

Note:

When you add an entire package to a deployment set, Deployer automatically also adds all ports associated with that package. Be sure to substitute configuration values for these ports if necessary (for instructions, see [“Substituting Configuration Values by Asset” on page 229](#)).

3. If you are done adding packages to the deployment set, go to [“Setting Package Properties” on page 39](#).

Adding Package Components

If you choose to add both package components and package files to a deployment set, you must be aware of the following:

- If you first select components, and then select files, Deployer only allows you to add files from the package file list.
- If you first select files, and then select components, Deployer might overwrite certain file selections to ensure consistency.

➤ To add package components

1. In the **Deployment Sets** area, under the deployment set to which to add package components, click the **Packages** folder. In the right-hand pane, Deployer lists all source Integration Servers.
2. In the right-hand pane, open the tree to show the packages on the source Integration Servers, and then click the name of a package that contains components to add to the deployment set.
3. In the **Select Components** area, open the tree to show the components in the package, select the check box next to each component to add to the deployment set, and then click **Save**.
4. Click **Return to Packages**. Deployer shows the partial package icon (📁) for the package in the left-hand pane under the **Packages** folder and in the right-hand pane, and a gray check mark for the package in the right-hand pane.
5. If you are done adding packages to the deployment set, go to [“Setting Package Properties” on page 39](#).

Manually Adding Dependencies to a Package Component in an IS & TN Deployment Set

Deployer cannot always detect all dependencies. If you are aware that an asset in an IS & TN deployment set has a dependency on a package component, and Deployer has not detected this dependency, you can manually *add* that dependency.

Deployer will check for the referenced asset when you map the project to target Integration Servers, as it does when you use the **Exists** option to resolve an unresolved dependency. If Deployer does not find the asset, an icon alerts you during the mapping task. If you do not resolve the dependency at that time, Deployer will write a message about it to the simulation report and, if you do not resolve it at that time, to the deployment report.

➤ To manually add a dependency on a package component

1. Go to the **Resolved Dependencies** page as explained in the previous section.
2. Under the **Manually Add Dependency** area, in the **Referenced Package** box, type the name of the package that contains the referenced component.
3. In the **Referenced Component** box, type the name of the referenced component.
4. Click **Add**.

You can remove a dependency you added manually. To do so, return to the **Projects > project > Define** page, open the folder that contains the asset, navigate to the asset in the tree in the right-hand pane, cancel the selection of the asset by clearing the appropriate check box, and save the deployment set.

Adding Package Files

If you choose to add both package components and package files to a deployment set, you must be aware of the following:

- If you first select components, and then select files, Deployer only allows you to add files from the package file list.
- If you first select files, and then select components, Deployer might overwrite certain file selections to ensure consistency.

If you add a partial package of only selected files to a deployment set and the package already exists on target Integration Servers, you can have Deployer delete specified files from the existing package on the target Integration Servers after deployment. You might use this feature if the existing package contains a service that has been superseded. In this case, you would deploy the files that make up the new service and delete the files that make up the old service.

> To add package files

1. In the **Deployment Sets** area, under the deployment set to which to add package files, click the **Packages** folder. In the right-hand pane, Deployer lists all source Integration Servers.
2. In the right-hand pane, open the tree to show the packages on the source Integration Servers, then click the name of a package that contains files to add to the deployment set.
3. Click **Select Files**. Deployer lists all files in the package. Do one of the following:

To add...	Do this...
All the files in the list	Click All files .
Only files you select in the list	Click Selected Files , then press the CTRL key and click each file to <i>include</i> in the deployment set.

To add...	Do this...
	Note: The Select Files option is a link near the top of the right-hand pane.
Only files <i>other than</i> those you select in the list	Click All except selected files , then press the CTRL key and click each file to <i>exclude</i> from the deployment set.
All files in the list whose name contains a specified string	Click Files specified by filter , then type the string on which to match the files to <i>include</i> in the deployment set. You can use an asterisk (*) as a wildcard character (for example, *.java or *.class).
All files in the list whose name does <i>not</i> contain a specified string	Click All except files specified by filter , then type the string on which to match the files to <i>exclude</i> from the deployment set. You can use an asterisk (*) as a wildcard character (for example, *.java or *.class).

- If a package of the same name as this partial package already exists on one of the deployment set's target Integration Server2s, and the existing package contains files to delete after deployment, type the fully qualified names of the files to delete in the **Files to Delete from Target** box. Type each file name on its own line, and end each line with a semicolon (;). For example:

```
code/classes/wm/administratorResource/admin.class;
code/classes/wm/administratorResource/user.class;
ns/wm/administratorResource/
```

- Click **Save**.
- Click **Return to Packages**. Deployer shows the partial package icon (📁) for the package in the left-hand pane under the **Packages** folder and in the right-hand pane, and a gray check mark for the package in the right-hand pane.
- If you are done adding packages to the deployment set, go to [“Setting Package Properties” on page 39](#).

Setting Package Properties

You must set properties for each package you added to the deployment set.

➤ To set properties for a package

- In the **Deployment Sets** area, under the deployment set to which you added entire or partial packages, open the tree under the **Packages** folder and click a package.
- In the **package_name Properties** area, specify the following properties:

Property	Entry						
Package Type	Use this property when the source package already exists on the target Integration Servers. You can use the following options for entire packages and for partial packages.						
	<table border="1"> <thead> <tr> <th>If you want Deployer to...</th> <th>Click...</th> </tr> </thead> <tbody> <tr> <td>Deploy the source package, replacing the existing package entirely. When you choose to deploy an entire package, this is the default.</td> <td>Full</td> </tr> <tr> <td>Deploy the components and files in the source package over the corresponding components and files in the existing package. When you choose to deploy package components, package files, or both, this is the default.</td> <td>Patch</td> </tr> </tbody> </table>	If you want Deployer to...	Click...	Deploy the source package, replacing the existing package entirely. When you choose to deploy an entire package, this is the default.	Full	Deploy the components and files in the source package over the corresponding components and files in the existing package. When you choose to deploy package components, package files, or both, this is the default.	Patch
	If you want Deployer to...	Click...					
Deploy the source package, replacing the existing package entirely. When you choose to deploy an entire package, this is the default.	Full						
Deploy the components and files in the source package over the corresponding components and files in the existing package. When you choose to deploy package components, package files, or both, this is the default.	Patch						
<p>Note: Before you deploy a project, you can find out which assets Deployer will overwrite by generating the simulation report.</p>							
Version	Supply the version number to use for the source package in comparisons with existing packages on target Integration Servers.						
	<p>Whether Deployer actually deploys the package depends on the version numbers of the source package and the existing package. If the source package's version number is the same or higher than the existing package's version number, Deployer deploys. If the source package's version number is lower than the existing package's version number, Deployer does not deploy.</p> <p>Note: The version number for the source package on the source Integration Server is not affected by your entry here.</p>						
Build	Supply the build number to assign to the deployed package on the target Integration Servers.						
	<p>Note: To retain the patch history of a package on the target server, you must specify a build number for the package.</p>						
Patches Included	Supply the list of patches that have been applied to the deployed package on the target Integration Servers. Specify the patch numbers, separated by commas (for example, 44, 45, 55). Specify patches only if you selected Full for Package Type .						
Brief Description	Supply a description to use for the deployed package on the target Integration Servers (for example, "December 2003 release with patches						

Property	Entry
	to correct Order Process problem.") Specify a description only if you selected Full for Package Type .

- In the **Recommendations for Target** area, you can recommend the minimum version of Integration Server and Java Virtual Machine (JVM) to run the source package. If the major version of the JVM on the target Integration Server is lower than you specify here, Deployer will deploy the source package but will not activate it, regardless of the setting of the **Activate After Deployment** option. When this happens, the target Integration Server will display a warning about the JVM version. The defaults shown in this area reflect the Integration Server and JVM that host the source package.
- In the **Package Build Options** area, indicate whether Deployer should use the package version and build numbers that exist in the source Integration Server each time the user creates a build instead of the package version and build numbers specified in the package_name **Properties** area.
- In the **Package Deployment Options** area, specify the following:

Option	Entry
Activate After Deployment	How Deployer should deploy the package. Click: <ul style="list-style-type: none"> ■ Activate to enable the package. ■ Install Only to install the package but not enable it. ■ Inbound Only to neither install nor enable the package.
Sync Document Types	Whether Deployer should synchronize the publishable IS document types in the source package with document types on the webMethods Brokers or Software AG Universal Messaging servers that are connected to the target Integration Servers.

Note:

The connected webMethods Brokers or Software AG Universal Messaging servers must be available at deployment time for synchronization to occur. If a connected webMethods Broker or Software AG Universal Messaging server is not available, IS document types are not synchronized for the Integration Server to which the webMethods Broker or Software AG Universal Messaging server is connected. Deployer writes a message to that effect to the deployment report. Deployer can detect webMethods Broker or Software AG Universal Messaging server unavailability when you generate the simulation

Option	Entry								
	report and will write a message advising you of the problem to the report.								
	<table border="1"> <thead> <tr> <th data-bbox="584 346 1153 399">To...</th> <th data-bbox="1153 346 1383 399">Select...</th> </tr> </thead> <tbody> <tr> <td data-bbox="584 399 1153 682">Synchronize all publishable IS document types in the package that are new to the target Integration Servers. Do not synchronize IS document types in the package that already exist on the target Integration Servers, even if they have been modified.</td> <td data-bbox="1153 399 1383 682">New</td> </tr> <tr> <td data-bbox="584 682 1153 777">Synchronize all publishable IS document types in the package.</td> <td data-bbox="1153 682 1383 777">All</td> </tr> <tr> <td data-bbox="584 777 1153 840">Do not synchronize any IS document types.</td> <td data-bbox="1153 777 1383 840">None</td> </tr> </tbody> </table>	To...	Select...	Synchronize all publishable IS document types in the package that are new to the target Integration Servers. Do not synchronize IS document types in the package that already exist on the target Integration Servers, even if they have been modified.	New	Synchronize all publishable IS document types in the package.	All	Do not synchronize any IS document types.	None
To...	Select...								
Synchronize all publishable IS document types in the package that are new to the target Integration Servers. Do not synchronize IS document types in the package that already exist on the target Integration Servers, even if they have been modified.	New								
Synchronize all publishable IS document types in the package.	All								
Do not synchronize any IS document types.	None								
Delete Universal Messaging Durables	<p>Controls whether Deployer deletes durables from Universal Messaging (UM) deployment sets automatically. To see this option, set Package Type to Full and Action After Deployment to Activate. When Package Type and Action After Deployment are set to any other value, this option is not available. Valid values are:</p> <ul style="list-style-type: none"> ■ Yes - deletes the UM durables. ■ No (Default) - does not delete the UM durables. 								

6. If you indicated in the project properties that you want Deployer to suspend individual triggers during deployment, click **Suspend Triggers**, select the check box next to each trigger to suspend, click **Suspend**, and then return to the previous page.
7. If you indicated in the project properties that you want Deployer to suspend individual adapter notifications during deployment, click **Suspend Notifications**, select the check box next to each notification to suspend, click **Suspend**, and then return to the previous page.

Note:

If you suspend a particular adapter notification but the notification does not exist on a target Integration Server, you will not be able to deploy. You can only suspend notifications that already exist on all target Integration Servers.

8. Click **Save**.
9. Repeat these steps for each package in the deployment set.

10. If you are not going to add any more assets to the deployment set, go to [“Resolving Dependencies” on page 45](#).

Adding webMethods Files

When Deployer deploys a webMethods file, the file retains the read/write permissions it had on the source server.

For large projects, you might be able to stream webMethods files from the source server to Deployer and from Deployer to the target server. For detailed information on this option, see [“Settings for Runtime-Based Deployment Projects” on page 202](#).

➤ To add Integration Server files

1. In the **Deployment Sets** area, under the deployment set to which to add webMethods files, click the **webMethodsFiles** folder. In the right-hand pane, Deployer lists the source Integration Servers you identified.
2. In the right-hand pane, open the tree to show the webMethods installation directory and its contents on the source Integration Servers. Select the check box next to each file to add to the deployment set.
3. Click **Save**. Deployer shows the selected assets in the left-hand pane under the **webMethodsFiles** folder for the deployment set.
4. If you are not going to add any more assets to the deployment set, go to [“Resolving Dependencies” on page 45](#).

Adding Trading Networks Assets

Note:

Your source and target Integration Servers might have assets in common, and you therefore do not need to deploy them from one environment to another. You can improve Deployer performance by excluding these assets from deployment sets. For instructions, see [“Excluding Common Assets” on page 44](#).

➤ To add Trading Networks assets

1. In the **Deployment Sets** area, under the deployment set to which to add Trading Networks assets, click the **Trading Networks** folder. In the right-hand pane, Deployer lists the source Integration Servers you identified.
2. In the right-hand pane, open the tree to show the Trading Networks assets on the source Integration Servers, then select the check box next to each asset to add to the deployment set.

Note:

If you add a TN document type that is set up in Trading Networks for duplicate checking using custom services, Deployer does not detect the dependency on the service. If the service does not already exist on the target Integration Servers, you must add the service to the deployment set. If you do not, Deployer will log an error to the deployment report and will not deploy the TN document type.

3. Click **Save**. Deployer shows the selected assets in the left-hand pane under the **Trading Networks** folder for the deployment set.
4. If you are not going to add any more assets to the deployment set, go to [“Resolving Dependencies” on page 45](#).

Excluding Common Assets

Your source and target Integration Servers might have assets in common, and therefore you do not need to deploy them from one environment to another. For example, you might have created certain packages that exist on all your source and target Integration Servers.

Deployer lets you identify these common assets so they will not appear in the asset list when you define deployment sets or as referenced assets when you resolve dependencies (see [“Resolving Dependencies” on page 45](#)). The assets you specify on this list will be excluded for all IS & TN deployment sets in all projects. Excluding these common assets improves Deployer performance by reducing the amount of processing needed to produce the asset and dependencies lists, and by preventing you from deploying unnecessary assets.

You identify the common assets in a file named `common.cnf` in the `Integration Server_directory \WmDeployer\config` directory. By default, the file is pre-populated with the names of Integration Server packages you should never deploy to other Integration Servers using Deployer, but rather should only install on other Integration Servers using the Software AG Installer. The file also includes instructions, lists the asset types you can exclude, and shows examples. List the assets you want to exclude next to the appropriate asset types.

Deploying ACLs through a Deployment Set

Deploying ACLs associated with My webMethods Server Groups

If you want to deploy ACLs that are associated with My webMethods Server groups, you must perform the following tasks.

1. Create an MWS deployment set to deploy the My webMethods Server groups to the target Integration Servers.
2. Create an IS & TN deployment set containing the ACLs associated with the My webMethods Server groups.
3. Mark the unresolved dependencies for the My webMethods Server groups as **Exists**.
4. Deploy the ACLs.

Deploying ACLs associated with LDAP Groups





If you want to deploy ACLs that are associated with LDAP groups, perform the following tasks.

1. Configure the LDAP groups on the target Integration Servers.
2. Create an IS & TN deployment set containing the ACLs associated with the LDAP Groups.
3. Mark the unresolved dependencies for the LDAP groups as **Exists**.
4. Deploy the ACLs.

Resolving Dependencies

Deployer can determine when assets that are in a deployment set require other assets that are not in the deployment set. The assets that require other assets are called *dependent* assets, while the assets that are required are called *referenced* assets. Deployer identifies missing referenced assets as *unresolved dependencies*.

Deployment Set	Example of Unresolved Dependencies
webMethods Broker	If you add a client group but not the documents to which the client group can publish or subscribe, the documents are unresolved dependencies.
IS & TN	If you add a trigger but not the service that is invoked by the trigger, the service is an unresolved dependency.
MWS	If you add a page but not the portlets that are referenced by the page, the portlets are unresolved dependencies.
Optimize	If you add a rule but not the dimensions used by the rule, the dimensions are unresolved dependencies.
ProcessModel	If you add a process model but not the flow services called by the process model, the flow services are unresolved dependencies.

In the project properties (“[Settings for Runtime-Based Deployment Projects](#)” on page 202), you indicated how you want to check dependencies in the deployment sets. When Deployer automatically checks dependencies and finds unresolved dependencies in a deployment set, it shows  in the **Unresolved Dependencies** column for the deployment set; when there are no unresolved dependencies, Deployer shows  in the column. When you can check dependencies manually, Deployer shows  in the **Unresolved Dependencies** column for each deployment set; click **Check** next to the . If necessary, you can later “un-resolve” or remove a dependency you have resolved and resolve it again a different way.

> To resolve dependencies

1. In the **Unresolved Dependencies** column for the deployment set, click **Check**. Deployer shows all unresolved dependencies on the **Unresolved Dependencies** page. The **Referenced**

Assets column lists the missing referenced assets. The next column offers the possible ways you can resolve the unresolved dependency. The **Asset** column shows the dependent assets.

2. Tell Deployer how to resolve each unresolved dependency. If you want to resolve all assets in a folder the same way, you can set the resolution at the folder level rather than at the level of the individual assets.

Option	Description
Add	If the referenced asset does not exist on the target servers and you want to deploy the referenced asset to them, use this option. Deployer adds the referenced asset to the deployment set. For Integration Server assets, you can choose to add the referenced asset or the entire package that contains the referenced asset.
Exists	<p>If you believe the referenced asset already exists on the target servers and you want to continue working, but you want Deployer to make sure the asset does in fact exist later, use this option. Deployer will check for the referenced asset when you map the project to target servers. If Deployer does not find the asset, an icon alerts you during the mapping task.</p> <p>If you do not address the problem during the mapping task, Deployer will write a message about the problem to the simulation report. If you deploy without addressing the problem, Deployer will not deploy the deployment set.</p>
Ignore	<p>If you want to bypass dependency checking for the referenced asset at this time so you can continue working, use this option. You might use this option if the referenced asset is missing on the source server. Missing referenced assets are marked with a question mark (?) on the Unresolved Dependencies page.</p> <p>Before deploying, make sure either that the referenced asset exists on the target server or that the referenced asset is unnecessary. If the referenced asset does not exist on the target server, Deployer might not be able to deploy correctly; if it can deploy, the deployed assets will not run correctly.</p> <p>Deployer will list ignored assets in the simulation report and in the deployment report.</p>
Unset	If you have set the assets in a folder to various settings and want to start over, use this option.

3. Click **Save**. Deployer moves dependencies you resolved using the **Exists** or **Ignore** option to the **Resolved Dependencies** page.
4. To see the resolved dependencies, click **Resolved Dependencies**.

You can un-resolve a resolved dependency and re-resolve it differently. To un-resolve a dependency, go to the **Resolved Dependencies** page, select the check box in the **Delete**

column for the resolved dependency, and click **Delete**. Deployer returns the dependency to the **Unresolved Dependencies** page. Go to that page and re-resolve the dependency.

Using Deletion Sets



You use *deletion sets* to identify those assets you want to delete from the target server during deployment. When you deploy the project, Deployer deletes the assets defined in the deletion set from the target server and then deploys assets defined for the deployment set from the source server to the target server. If Deployer does not display a server you want to use as a target, you have not yet set it up to work with Deployer. After you create a deletion set, you should identify the target servers from which to delete assets and then add the assets you want to delete in the deletion set.

You can also export deletion set definitions from one project and import them into another.

For runtime-based deployment, Deployer supports deletion sets for Broker, Integration Server, and Trading Networks assets only.

Creating a Deletion Set

➤ To create a deletion set

1. In Deployer, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. In the right-hand pane, click  **Define**.
5. Click **Create Set**.

Deployer displays the Create Set properties in the right-hand pane.

6. Complete the following fields:

Box	Entry
Type	Runtime type of the server that contains the assets to delete.
Set	Deletion
Name	Name to use for the deletion set. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "

Box	Entry
Description	Description for the deletion set. The description length has no limit and can include any characters.
Maximum TN Assets to Display	Number of Trading Networks assets Deployer should display. Depending on your browser, Deployer might not be able to display more than 1000 assets for Trading Networks. If the source server hosts more than 1000 assets, use this field with the with the All other assets field to reduce the number of assets displayed.
Packages (IS & TN deletion set)	After you choose the servers from which to define deletion sets, Deployer displays all packages on the servers. You can use this field to narrow the display. Type a regular expression that specifies the text that the package names must contain in order to be listed.
All other assets	After you choose the servers from which to define deletion sets, Deployer will display all assets on those servers. You can use this field to narrow the display. Specify a regular expression that specifies the text that the asset names must contain in order to be listed.
	<p>Note: Deployer can display up to 10,000 assets. If the source server hosts more than 10,000 assets, use the Packages and All other Assets fields to reduce the number of assets to be displayed.</p>

7. Click **Create**.

Deployer displays the deletion set in the left-hand pane in the **Deletion Sets** area.

8. Perform the tasks in [“Identifying Servers for a Runtime-Based Deletion Set”](#) on page 48 to identify the servers that contain the assets to add to the deletion set.

Identifying Servers for a Runtime-Based Deletion Set

Deployer lists the version of each target server in the **Version** column. You must select target servers with the same version for the deletion set. You cannot include target servers with different versions in a deletion set.

> To identify servers for a runtime-based project

1. On the **Deployer > Projects > project > Define** page, in the left-hand pane in the **Name** column, click the deletion set for which to identify servers.
2. In the **Select Servers** column in the right-hand pane, select the check box next to each server that contains assets to add to the deletion set.

3. Click **Save**.
4. Perform the tasks in [“Adding Assets to a Runtime-Based Deletion Set”](#) on page 49 to add assets to the deletion set.

Adding Assets to a Runtime-Based Deletion Set

You can choose assets to add to a deletion set from any server that is similar to the target server from which you want to actually delete the assets. For Trading Networks, you cannot delete document attributes, field definitions, binary types, or profile security data.

➤ To add assets to a runtime-based deletion set

1. In the **Deletion Sets** area, under the deletion set to which to add assets, click the folder that contains assets you want to add to the deletion set.

In the right-hand pane, Deployer lists the servers you identified as target servers from which to delete assets.

Note:

For information about adding full or partial packages to a deletion set, see [“Adding Packages to Deletion Sets”](#) on page 49.

2. In the **Select** column of the **Select Servers** area, select the check box next to each source server that contains assets to add to the deployment set.

Deployer lists the version of each source server in the **Version** column. You must select servers with the same version for the deletion set. You cannot include servers with different versions in a deletion set.

3. In the right-hand pane, open the tree to show the assets on the servers, select the check box next to each asset to add to the deletion set, and then click **Save**.

Deployer shows the selected assets in the left-hand pane under the folder you clicked in the previous step.


Adding Packages to Deletion Sets

If you are creating an IS & TN deletion set, you can add Integration Server packages in their entirety, or you can add selected package components only (called *partial packages*). You can add full or partial packages to either runtime or repository-based projects.

➤ To add full or partial packages to deletion sets





1. Perform one of the following:

If your project is... Do this...

Runtime-based	In the Deletion Sets Runtime area, under the deletion set to which to add packages or partial packages, click the Packages folder.
Repository-based	In the Deletion Sets area, under the deletion set to which to add the packages, click the server  that contains packages or partial packages.

In the right-hand pane, Deployer lists the Integration Servers you identified in [“Identifying Servers for a Runtime-Based Deletion Set”](#) on page 48.

- In the right-hand pane, expand the tree to display the packages on the Integration Servers, then do one of the following:

To add...	Do this...
Full packages	Select the check boxes next to the packages to add in their entirety to the deletion set and click Save . Deployer shows the entire package icon  for the selected packages in the left-hand pane under the Packages folder (for runtime-based deployment) or under the server  (for repository-based deployment).
Partial packages	<ol style="list-style-type: none"> Click the package name. In the Select Components area, open the tree to show the package components, then select the check box next to each component to add to the deletion set. Click Save. Click Return to Packages. Deployer shows the partial package icon  for the package in the left-hand pane under the Packages folder (for runtime-based deployment) or under the server  (for repository-based deployment).

Note:

If you add a partial package and later want to include the entire package, cancel the selection of the components by clicking the name of the partial package, clearing all checked boxes, and clicking **Save**. Then save the deletion set and add the entire package.

Exporting and Importing Deletion Set Definitions



For runtime-based projects, Deployer exports and imports deletion set definitions separately from the rest of the project properties exported and imported in [“Exporting and Importing Project Properties”](#) on page 212.

Note:

You cannot export and import deletion set definitions for repository-based projects.


When you export deletion set definitions, Deployer creates a file called *project_deleteSets.xml* that contains the deletion set definitions. This file is stored in the *Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \outbound* directory. Deployer also gives you the option to save the file to your local file system. You can then use this file to import the deletion set definitions into another Deployer project.


➤ To export and import deletion set definitions



1. Export deletion set definitions from a project as follows:
 - a. In Deployer, go to the **Deployer > Projects** page.
 - b. In the **Name** column, click the project from which to export.
 - c. In the right-hand pane, click  **Define**.
 - d. Click **Export Deletion Set Definitions**.
2. Import deletion set definitions into a project as follows:
 - a. Copy the *project_deleteSets.xml* file to the *Integration Server_directory \instances \instance_name \packages \ WmDeployer \replicate \inbound* directory on the machine that hosts the project.
 - b. In Deployer, go to the **Deployer > Projects** page.
 - c. In the **Name** column, click the project into which to import.
 - d. In the right-hand pane, click  **Define**.
 - e. Click **Import Deletion Set Definitions**, then select the *project_deleteSets.xml* file you just copied to the inbound directory.


Creating a Build

➤ To create a build

1. In Deployer, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.

- Click  **Build**. Deployer displays the **Projects > project > Build** page and lists all builds that exist for the selected project.

The **Status** column on the **Projects > project > Build** page indicates whether each project build is in sync with the current project definition. If the build and the current project definition are in sync, the column shows . If the project definition has changed since the build was created, the column shows . You can rebuild such a project if you want. For instructions, see [“Rebuilding a Build” on page 53](#).

To see the progress report of the current or last action, click  in the **Progress Report** column. The progress report displays the updates for build requests as they occur. This is useful in the case where the deployment build takes a long time to finish.


- In the left-hand pane, click **Create Build**.
- In the **Name** box accept the default build name or replace it with a name that you choose. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:

\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "

- In the **Description** box, you can type a description for the build. The description can be of any length and can include any characters.
- Click **Create**.



Note:


If the project for which you are trying to create the build contains unresolved dependencies, you will receive a message to that effect and the build process will fail. For instructions on displaying and resolving unresolved dependencies, see [“Resolving Dependencies” on page 45](#).


- To view the progress of the build, click the **View Progress Report** link. The progress report displays the updates for build requests as they occur. This is useful in the case where the deployment build takes a long time to finish.
- Under **Build History** in the right-hand pane, click  in the **Report** column to display the build report in HTML or XML.

The build report lists the assets that were successfully included in the build, describes any errors that occurred during the build process, and informs you if the project contains unresolved dependencies. The report is also available under the name `BuildReport_reportID.xml` in the `Integration Server_directory \instances\instance_name\packages\WmDeployer\pub\projects\project_name\builds\build_name\reports` folder, where `project_name` is the name of the project and `build_name` is the name of the build.

Rebuilding a Build

The **Status** column on the **Projects > project > Build** page indicates whether each project build is in sync with the current project definition. If the build and the current project definition are in sync, the column shows . If the project definition has changed since the build was created, the column shows .

If a project build is out of sync with the current project definition or contains assets that you know have changed on the source servers, and you want to re-create the build to bring it up to date, click  in the **Rebuild** column for the build.

If you want to see the progress report, click  in the **Progress Report** column. The progress report displays the updates for build and rebuild requests as they occur. This is useful in the case where the deployment build is large and it takes a long time to finish.

Note:


Deployer does not take dependencies into account while rebuilding project builds.

If the project for which you are trying to create the build contains unresolved dependencies, you will receive a message to that effect and the build process will fail. When asset dependencies are changed on the source server, you must first remove the dependent asset from the deployment set and save the deployment set. Then, add the asset back and save the deployment set. When Deployer displays the changed dependencies, resolve the dependencies and rebuild the project. For instructions on displaying and resolving unresolved dependencies, see [“Resolving Dependencies” on page 45](#).

Exporting and Importing a Build

You can export the build you want to deploy from the Deployer in one environment and import the build into the Deployer in another environment. The Deployer into which you import the build automatically creates the deployment project and deployment sets from the imported build. You can then map the imported build, or you can export a deployment map for the build from the Deployer in the source environment and import it into the target project. For more information about importing and exporting maps, see [“Exporting and Importing a Map” on page 228](#).

> To export and import a build

1. Export a build as follows:
 - a. In the source Deployer, go to the **Deployer > Projects > project > Build** page.
 - b. Locate the build to export and click  in the build's **Export** column. Deployer creates a file that contains the build. The file is named `projectName_ExportedBuild_buildName` and is stored in the `Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \outbound` directory. Deployer also allows you to save the file to your local file system.

- c. If you have previously exported a build of the same name, Deployer displays a dialog box confirming that you want to overwrite the existing build. Click **OK** to overwrite the existing build.

2. Import the build as follows:

Note:

You can import builds that are exported from an older version of Deployer to a newer version of Deployer. But the assets present in the builds which are exported from an older version of Deployer may or may not get deployed. Even if the assets are deployed, the deployed assets may or may not function as expected.

- a. Copy the *projectName_ExportedBuild_buildName* file to the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\inbound directory on the machine that hosts the target Deployer.
- b. In the target Deployer, go to the **Tools > Import Build** page.
- c. In the **Project Build** list, click the *projectName_ExportedBuild_buildName* file you just copied to the inbound directory.
- d. Click **Import**.

4 Deploying Assets from a Source Asset Repository

■ About Repository-Based Deployment	56
■ Building Composites	57
■ Connecting to a Source Repository	69
■ Connecting to Target webMethods Servers	70
■ Creating Repository Deployment Projects	87
■ Creating a Deployment Set from Repository Assets	88
■ Identifying the Source Repository for a Deployment Set	88
■ Selecting Composites	89
■ Selecting Individual Assets from Composites	89
■ Deploying ACLs through a Deployment Set	90
■ Resolving Dependencies	91
■ Resolving Conflicts	94
■ Using Deletion Sets	95

About Repository-Based Deployment

In repository-based deployment, you use the Asset Build Environment to build the assets from your development environment or version control system (VCS) into composites and store them in a *repository*. The repository is a specific directory structure to which you map Deployer. Deployer deploys the assets contained in the repository to target servers, target groups, or both.

You can use repository-based deployment to deploy assets for the following webMethods run-time components to target servers:

- ActiveTransfer
- AgileApps
- API Gateway
- Application Platform
- Process Engine (BPM Process Development)
- Broker
- Business Rules
- Digital Event Services
- Event Routing
- Event Server (supported only for Event Server version 9.5 or lower)
- Integration Server
- My webMethods Server
- Optimize
- Trading Networks
- Task Engine
- Universal Messaging

What is the Asset Build Environment?

To build composites for deployment, you must first install the webMethods Asset Build Environment. The Asset Build Environment supplies the scripts necessary to build assets into composites that Deployer then deploys to target servers and groups. For testing purposes, you can install the Asset Build Environment on the machine that you use to create assets in the development environment. For daily or continuous builds, you should install the Asset Build Environment on a separate build machine that you use to build the assets into composite files that Deployer can deploy to target servers. For information about installing the Asset Build Environment, see *Installing Software AG Products*.

After installing the Asset Build Environment, you can locate the scripts and properties files for the Asset Build Environment in the following directory:

Software AG_directory \common\AssetBuildEnvironment\master_build

- **build.properties** Contains the settings that the build script uses to build the assets. You must set the properties for the build before running the build script.
- **build-number.xml** The build script uses this file to automatically version the assets included in the build incrementally. You can set a custom build number in this file (for example, to match the build number to the VCS revision number).
- **build-source-checkout.xml** If the build source directory is a version control system, you can use this file as a template to create an Ant task that checks out sources from a version control system (VCS).
- **build.bat or build.sh** The build script files for Windows and Unix systems.

When you upgrade the Asset Build Environment to a higher release version using the *overinstall* installation method, any changes you made to these files are retained. You should move the files only when you uninstall and then create a new installation of the Asset Build Environment.

What are Composites?

Composites are compressed files that contain the definitions of the assets you created in a development environment and their dependencies. Each composite defines assets from one webMethods run-time component. You build the assets into the composite file during the build process and store them in a repository for deployment.

Deployer can use the assets from several different composites (and webMethods run-time components) to construct a deployment set. The assets in each composite file retain the same relationships they share in the development environment.

The build process that creates the composites for your assets also creates an *Asset Composite Definition Language descriptor* (ACDL descriptor) for each composite. The ACDL descriptor is an XML schema that serves as a manifest for each composite and describes all of the assets included in the composite file. Deployer reads the descriptor to determine which assets are present in each composite.

Building Composites

The build process for composites involves the following high-level tasks:

1. Create the assets on the development environment and export or copy them to a file system or version control system (VCS), from which the build process can obtain them. In this help, the location to which you export the assets is called the *source directory*. You can find the information about how to prepare, extract, or export the assets to the source directory in the documentation of the webMethods product(s) for which you want to deploy assets.
2. Install the webMethods Asset Build Environment on the build machine as described in *Installing Software AG Products*.

3. Modify the `build.properties` file to define the parameters used by the master build script to build composites.
4. Run the master build script to package your assets into composites for deployment.

Adding Assets to the Source Directory

You create assets with a webMethods runtime and store them in a file system or version control system (VCS). The directory of the file system or VCS in which you store the assets is the *source directory*.

Most webMethods runtimes supply a means of exporting or saving the assets you create to the source directory. You can find details about how a product exports assets to the source directory in the documentation of the respective product. However, to add *most* of the Integration Server administrative assets to the source directory, you must manually copy or check in the related configuration folders to the source directory. For a complete list of files and directories to copy or check in to the source directory for Integration Server administrative assets, see [“Adding Administrative Assets to the Source Directory” on page 127](#).

When specifying the properties for the `build.properties` file, you use the `build.source.dir` parameter to specify the full path of each source directory. When you run the build script, it packages the contents of the source directories into the correct composite file format for deploying those assets to target servers.

Keep the following points in mind when adding assets to the source directory:

- The build script must have access to the source directory.
- If the source directory is a file system on a VCS, you must supply the proper credentials for accessing and checking out the composite files from the source directory. For more information about supplying the VCS checkout properties, see [“Setting VCS Checkout Properties” on page 64](#).

Setting Build Properties

The `build.properties` file controls the build settings for the repository-based build process. The file contains a set of switches that enable and disable the build tasks for the corresponding runtimes in the build.

> To set the build properties

1. Open the following file in a text editor:

```
Software AG_directory \common\AssetBuildEnvironment\master_build\build.properties
```

2. Set the following build properties:

- `sag.install.dir`

The path to the installation directory of the Software AG products.

- `build.source.dir`

A list with the full paths to the source directories that contain assets to build. Use a forward slash "/" as path separator. All source directories in the list must contain project directories (such as IS packages, CAF projects, and TN exports) as direct children. Use ";" as the delimiter when listing more than one path. For example:

```
/<root_path>/<project_name>/IS;/<root_path>/<project_name>/TN
```

Where `<root_path>` is the root directory of the source, `<project_name>/IS` is the project directory holding Integration Server assets, and `<project_name>/TN` is the project directory holding Trading Networks assets.

Most products do not have restrictions or a naming convention for the source directory that contains their assets, except the products in the following table:

Product name	Naming convention or restrictions for the source directory
AgileApps Cloud	AgileAppsCloud
API Gateway	APIGateway
Broker	Broker
Integration Server	"Adding Administrative Assets to the Source Directory" on page 127 "Adding Package Assets to the Source Directory" on page 170
Universal Messaging	UniversalMessaging

- `build.output.dir`

The root directory in which the build script will place the output (composites and descriptors) of the build. Use a forward slash "/" as path separator. For example:

```
build.output.dir=<root_path>/build/
```

In the output directory, the build script creates a subdirectory with the current build number and a subdirectory for the composites and descriptors for each runtime type included in the build. The following table lists the name of the subdirectory that the build creates for each product:

For this product	The build creates a subdirectory with name
ActiveTransfer	MFT
AgileApps Cloud	AgileAppsCloud
API Gateway	APIGateway

For this product	The build creates a subdirectory with name
Application Platform	ApplicationPlatform
BPM Process Development	BPM
Broker	Broker
Business Rules	Rules
Digital Event Services	DES
Event Routing	EDA
Integration Server	IS
My webMethods Server	MWS
Optimize	Optimize
Trading Networks	TN
Task Engine	TaskEngine
Universal Messaging	UniversalMessaging

- `build.source.project.dir`

A list of paths to the project directories that contain the assets to build. Use a forward slash "/" as path separator. This property is different from `build.source.dir`, because you use it to define individual project directories. Use ";" as the delimiter when listing more than one path.

- `enable.build.ProductID`

With this property, you can enable or disable building the composites for specific runtimes included in the build. Valid values are `true` (build composites for the specified runtime) and `false` (exclude the specified runtime from the build). Each runtime has a separate property that you must specify on a separate line, for example:

```
enable.build.UniversalMessaging=true
enable.build.IS=true
```

The following sample lists the enable build property for each runtime:

```
enable.build.MFT=           # ActiveTransfer
enable.build.AgileApps=     # AgileApps Cloud
enable.build.APIGateway=
enable.build.ApplicationPlatform=
enable.build.IS=           # Integration Server
enable.build.MWS=         # My webMethods Server. Set to "false" if
building Task Engine assets.
enable.build.BPM=         # BPM Process Development
enable.build.TN=         # Trading Networks
enable.build.TaskEngine=   # Set to "false" if building My webMethods
Server assets.
```

```

enable.build.Optimize=
enable.build.Broker=
enable.build.DES=           # Digital Event Services
enable.build.EDA=          # Event Routing
enable.build.Rules=        # Business Rules
enable.build.UniversalMessaging=

```

- `enable.archive`

Specifies whether the build script archives the output directory. Valid values are `true` (create an archive) and `false` (do not create an archive). If you set this field to `true`, you must provide a value for `build.archive.dir`.

- `build.archive.dir`

Required if `enable.archive` is set to `true`. Path to the archive directory for the output. The build script creates a new subdirectory with the name of the build number and moves the contents of the output directory (set in `build.output.dir`) to the archive directory.

- `build.version`

Version number of the current build. The build script appends an automatically-generated incremental build number to this property to get the final build number. For example, if you set this property to `9.12`, the generated build number for the first build is `9.12.1` and the build number for the next build is `9.12.2`.

- `enable.checkout`

Specifies whether the build script should check out the source files from a VCS. Valid values are `true` (enables the check-out task) and `false` (disables the check-out task). When the checkout is enabled, the build script invokes the default target you set in the `build-source-checkout.xml` file. For details about configuring the properties in `build-source-checkout.xml` to synchronize with your VCS system, see [“Setting VCS Checkout Properties” on page 64](#).

- `build.log.enable`

Specifies whether to enable logging for the build. Valid values are `true` (enable logging) and `false` (disable logging).

- `build.log.fileName`

The name of the log file. The default file name is `build.log`. The build script creates the log file in the following directory:

```
Software AG_directory \common\AssetBuildEnvironment\bin
```

- `build.logLevel`

The logging level for the messages recorded in the log. Valid values are:

- `debug` - Debug, informational, warning, error, and fatal messages.
- `error` - Error and fatal messages.
- `info` (default) - Informational, warning, error, and fatal messages.

- verbose - No possible values defined for log level.
 - warn - Warning, error, and fatal messages.
3. To push composites and descriptors to a Git repository, specify values for the Git properties listed in [“Setting Git Properties” on page 62](#).
 4. For some of the products, you must specify build properties specific to the product. Based on which products you want to include in the build, configure the product-specific build properties as described in:
 - [“API Gateway-Specific Build Properties” on page 108](#)
 - [“BPM-Specific Build Properties” on page 119](#)
 - [“Optimize-Specific Build Properties” on page 194](#)
 - [“Business Rules-Specific Build Properties” on page 124](#)
 - [“Integration Server-Specific Build Properties” on page 132](#)
 5. Save and close the file.

Setting Git Properties

To use Asset Build Environment to push composites and descriptors to a Git repository, in the `build.properties` file specify values for the Git properties listed in the following table:

Property	Definition
<code>localStore</code>	The location of the folder on the local file system, in which the Git repository is cloned or the local Git repository. <code>localStore</code> is the local Git repository that is the parent of the <code>build.output.dir</code> directory.
<code>remoteStore</code>	The URL of the remote Git repository.
<code>branch</code>	Optional. The name of the branch in the Git repository to which to push the composites and descriptors.
<code>message</code>	Required. A message that gives context and describes the changes included in the Git commit.
<code>username</code>	Required to connect over HTTPS. The name of a user that can log on to the remote Git repository.
<code>password</code>	Required to connect over HTTPS. The password for the specified username.
<code>pathToPrivateKey</code>	Required to connect over SSH. The path to the location on the file system in which you store your private key.

Property	Definition
<code>pathToPublicKey</code>	Required to connect over SSH. The path to the location on the file system in which you store your public key.
<code>privateKeyPassword</code>	Required to connect over SSH. The password for your private key.
<code>pathToKnownHosts</code>	Required to connect over SSH. The path to the location on the file system in which you store the known hosts.
<code>tag</code>	Optional. The name of the tag in the remote Git repository to create for the uploaded assets. If you do not specify <code>tag</code> , the script does not create a tag for the uploaded assets.
<code>fileCommand</code>	Optional. The name of the command to send to the remote Git repository. The only currently supported value is the command for variable substitution, <code>transform-properties</code> . If you do not specify <code>fileCommand</code> , the script does not send a command to the repository.
<code>registryRoot</code>	Optional. The location of the registry folder inside the Git repository relative to the repository folder. If you do not specify <code>registryRoot</code> , by default the registry root is resolved to "." (the same location as the repository root directory). If you specify a value for <code>registryRoot</code> that does not exist, the script throws an exception.

Setting JVM Memory

You can adjust the JVM memory settings available to the build. This is helpful when building a large number of assets or assets that contain a large amount of data.

➤ To change the JVM memory settings

- Using a text editor, open the following file:

For this platform...	Open the following file...
Windows	<i>Software AG_directory</i> \common\AssetBuildEnvironment\bin\build.bat
UNIX	<i>Software AG_directory</i> /common/AssetBuildEnvironment/bin/build.sh

- Change the values of the following parameters as required:

- `JAVA_MAX_MEM`
- `JAVA_MAX_PERM_SIZE`

3. Save and close the file.

Setting VCS Checkout Properties

If you use a VCS to store your assets, you must set the `enable.checkout` property in `build.properties` to `true` and configure the properties in `build-source-checkout.xml` to access the VCS. The `build-source-checkout.xml` template is specific to an SVN version control system. You might have to add more properties to make the file work with other VCS types.

> To set SVN checkout properties

1. Open the following file in a text editor:

```
Software AG_directory
\common\AssetBuildEnvironment\master_build\build-source-checkout.xml
```

2. Set the following properties to correspond to your SVN as follows:

Note:

Property	Definition
svn.jars.dir	Location of the open source SvnAnt libraries (http://subclipse.tigris.org/svnant.html). These libraries are required by the checkout Ant task. To use a VCS other than SVN, modify this parameter accordingly.
svn.user	User name of the SVN.
svn.password	Password of the SVN.
svn.url	URL of the SVN from where the build script checks out the assets.
build.checkout.dir	The root directory from which the build script will check out the asset sources. This property can point to only one directory; therefore, it should be the root directory containing <i>all</i> projects that will be built. For example, if your solution contains an Integration Server project, your full project directory might be <code>/root/project/builds/webM/IS</code> . In this case, you would not include <code>/IS</code> in the path. Instead, you would enter <code>/root/project/src/webM/</code> .

3. Save and close the file.

Note:

Do not rename or remove the file from the `master_build` directory.

Running the Build Script

After you have set the properties in the `build.properties` file, you can run the build script from the `Software AG_directory \common\AssetBuildEnvironment\bin` directory.

- To build composites and descriptors:

For this platform...	Run the following command...
Windows	<code>build.bat</code>
UNIX	<code>build.sh</code>

The build script places the composites and descriptor in the location you specified in the `build.output.dir` property in the `build.properties` file.

- To build and upload the composites and descriptors to a remote Git repository:

For this platform...	Run the following command...
Windows	<code>build.bat buildUploadAssets</code>
UNIX	<code>build.sh buildUploadAssets</code>

Note that `buildUploadAssets` supports both HTTPS and SSH. When configuring the Git properties in the `build.properties` file, based on the protocol used to connect to Git, you must specify:

- For HTTPS - the username and password of a user account that can access the Git repository over HTTPS.
- For SSH - values for the `<pathToPrivateKey>`, `<pathToPublicKey>`, `<privateKeyPassword>`, and `<pathToKnownHosts>` properties.

Build Script Processing Steps and Output Files

The build script goes through the following processing steps:

- Checks out the asset sources.
- Creates and indexes the repository.
- Versions the assets.
- Builds the composites and descriptors in the repository.

The build script uses the `build-number.xml` file to automatically version the assets included in the build in the format `version_number.build_number`.

The build script generates the following files for each runtime in the build:

- ACDL descriptor file in the format `project_name.acdl`

This is the descriptor file that contains the metadata that lists the assets and dependencies contained in the composite.

- A compressed composite file that contains the definitions of the assets and their dependencies for deployment. The composite file is in the format *project_name.composite_file_type* as described in the following table:

For this runtime type *project_name* is *composite_file_type* is

ActiveTransfer The name of the file you exported using the `wm.mft.admin:exportData` built-in service. For more information about `wm.mft.admin:exportData`, see *webMethods ActiveTransfer Built-In Services Reference*. .bin

API Gateway APIGatewayAssets.acdl .zip

Application Platform The name of the standard web application project. .war

The name of the OSGi bundle project. .jar

The name of the OSGi web bundle project. .jar

The difference between bundle and Web bundle is in the jar content. Web bundles are valid OSGi bundles and are structured internally as .war files, that is, they are a fusion of the first two project types. The difference is also reflected in the "acdl" file of the web bundles. They contain both an asset of type "Bundle" and an asset of type "WebApp".

Configuration composites that contain configuration files, with the ".properties" file extension. They do not have their own projects, but are co-hosted in the `/src/main/config/` directory of either a war, bundle, or web bundle project. The name of the host project is not used as the name of the configuration composite. The files are just copied into the asset repository with the name *property_file_name.properties*.

An additional *property_file_name.acdl* file is created for the configuration composite. You can create a project that has only a collection of configurations in `/src/main/config`. The configurations can customize either other composites, or the OSGi Common Platform.

BPM process model The process ID with "/" replaced by "_". For more information .process

For this runtime type	<i>project_name</i> is	<i>composite_file_type</i> is
	about process models, see <i>webMethods BPM Process Development Help</i> .	
Digital Event Services	The name of the folder that contains the DES "TypeRepository" folder.	.zip
Broker	The name of the file you exported using My webMethods Server prefixed with "Broker_" or "Provider_" depending on whether you exported Broker assets or JNDI assets, respectively. When you export Broker assets (document, client, or client group), Broker creates an ADL file for the Broker assets. To export assets from webMethods Broker 8.2 SP2 or earlier, you must migrate the assets from XML format to ADL format for use in Deployer. For more information about migrating Broker assets from XML to ADL format, see <i>Upgrading Software AG Products</i> . When you export JMS destinations (JMS queues and JMS topics) created by a webMethods JNDI provider, Broker creates a separate XML file for a JNDI context. For more information about exporting Broker assets, see <i>Administering webMethods Broker</i> .	.adl (Broker assets) .xml (JNDI assets)
Business Rules	The name of the rules project.	.jar
EDA	The name of the Event Types project.	.zip (Event Type assets)
Integration Server	■ For package, adapter runtime, and .Net assets, this is the name of the Integration Server package. For more information about adding package assets	.zip

For this runtime type	<i>project_name</i> is	<i>composite_file_type</i> is
	<p>to the source directory for inclusion in the build, see “Adding Package Assets to the Source Directory” on page 170. For more information about adding adapter runtime or .Net assets to the source directory for inclusion in the build, see “Adding Adapter Runtime and .NET Service Assets to the Source Directory” on page 175.</p> <ul style="list-style-type: none"> ■ For administrative assets, this is isconfiguration. For more information about adding administrative assets to the source directory for inclusion in the build, see “Adding Administrative Assets to the Source Directory” on page 127. 	
My webMethods Server	The name of the project you exported. For more information about exporting My webMethods Server assets, see <i>Administering My webMethods Server</i> .	.skin, .war, .cdp
AgileApps Cloud	The name of the application you exported using the AgileApps Cloud server. For more information about exporting AgileApps Cloud assets, see the AgileApps Cloud documentation.	.zip
Optimize	The name of the file you exported using My webMethods Server. For more information about exporting Optimize assets, see <i>Administering webMethods Optimize</i> .	.zip
Trading Networks	The name of the file you exported using My webMethods Server. For more information about exporting Trading Networks assets, see <i>webMethods Trading Networks Administrator’s Guide</i> .	.bin

For this runtime type	<i>project_name</i> is	<i>composite_file_type</i> is
Task Engine	The name of the task application project.	.war
Universal Messaging	The name of the file you exported using Universal Messaging Enterprise Manager.	.xml

Connecting to a Source Repository

You define a source repository as the source server, and specify the location of the repository directory from which the assets should be deployed.

➤ To define the source repository

1. In Deployer, go to the **Deployer > Repository** page.
2. Click **Add Repository**.

Deployer displays the repository properties in the right-hand pane.

3. In the **Name** field, type the name to use for the repository alias. The name has an upper limit of 32 characters and cannot contain spaces or the following special characters:


`$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "`

4. In the **File Directory** field, type the full path of the repository directory in which the composites are located.

Deployer must have access to the repository directory.

5. Click **Configure**.

Deployer displays the repository in the table in the **Repositories** pane.


6. To test the connection, click  in the **Test** column of the **Repositories** pane.
7. To rebuild the index, see [“Rebuilding the Index” on page 69](#).

Rebuilding the Index

If the index you created becomes corrupted or the repository index is accidentally deleted from the repository, you can recreate the index without rebuilding the entire repository.

When you follow this procedure to rebuild the index, Deployer rebuilds *only* the index. To rebuild the entire repository (including the index), you must run the build script as described in “[Running the Build Script](#)” on page 65.

➤ **To rebuild the index**

1. In Deployer, go to the **Deployer > Repository** page.
2. If you want to change the repository directory, perform the following:
 - a. In the **Name** column, click the name of the repository to edit.
Deployer opens the repository properties in the right-hand pane.
 - b. In the **File Directory** box, type the full path of the repository for which to rebuild the index.
 - c. Click **Save Changes**.
3. In the **Create Index** column for the repository, click .

Connecting to Target webMethods Servers

When deploying assets from a source repository, the *target* servers are the target runtimes to which Deployer deploys assets from the source repository.

The version of the target servers that you add in a project is determined as follows:

- Deployer limits the version of the target servers in your project based on the first target server or target group you select for mapping. For example, if you select a target server of version 9.12, Deployer displays only target servers and target groups of version 9.12 for mapping.
- Each project includes source repositories and targets servers of only one version. You cannot include source repositories or target servers of different versions.
- Deployer deploys only to target servers and target groups of version 8.2 SP1 and higher.

Connecting to a Target ActiveTransfer Server

Use the following procedure to set up connections to target ActiveTransfer Server instances.

➤ **To connect to target ActiveTransfer Server instances**

1. In Deployer, go to the **Servers > ActiveTransfer** page.
2. Click **Configure ActiveTransfer** and complete these fields:

Field	Entry
Name	Name to assign to the ActiveTransfer Server instance. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the ActiveTransfer Server instance.
Port	Port for the ActiveTransfer Server instance.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the ActiveTransfer Server instance. For information about selecting the version, see .
Use SSL	Whether Deployer should use SSL to connect to the server.

3. Click **Configure**. To test the connection, click .

Connecting to a Target API Gateway Server

Use the following procedure to set up connections to target API Gateway servers.

➤ To connect to target API Gateway servers

1. In Deployer, go to the **Servers > APIGateway** page.
2. For every target API Gateway server, click **Configure APIGateway Server**. In the **Configure APIGateway Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the Integration Server in which this API Gateway is running. Note: You cannot use "localhost" for API Gateway servers.
Port	Port of the Integration Server in which this API Gateway is running.

Field	Entry
User	User name for a user account with Administrator along with Import Assets and Export Assets Authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the API Gateway server. For information about selecting the version, see .
Use SSL	Whether Deployer should use SSL to connect to the server.

3. Click **Configure**. To test the connection, click .

Connecting to a Deployment Endpoint for Digital Event Services

The deployment endpoints for Digital Event Service (DES) are hosted by the Software AG Platform Manager server because DES is not a standalone server runtime. When you configure a server for DES asset type deployment, you provide the host name and port number of the Platform Manager that manages the target Software AG installation in which DES is installed.

> To connect to target DES endpoints

1. In Deployer, go to the **Servers > Digital Event Services** page.
2. For every target DES endpoint, click **Configure Digital Event Services Server**. In the **Configure Digital Event Services Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name of the Platform Manager that manages the target Software AG installation in which DES is embedded. Note: You cannot use "localhost" for DES servers.
Port	Port number of the Platform Manager that manages the target Software AG installation in which DES is embedded.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.

Field	Entry
Version	Version of the DES server. For information about selecting the version, see .
Use SSL	Whether Deployer should use SSL to connect to the server.

- Click **Configure**. To test the connection, click .

Connecting to a Target Integration Server and Trading Networks

All connections you create for Integration Servers and Trading Networks servers are *remote server aliases*. A remote server alias contains the connection information required to connect to a remote Integration Server or Trading Networks server. For more information about remote servers, and instructions on defining them, see *webMethods Integration Server Administrator's Guide*.

➤ To connect to target Integration Servers and Trading Networks servers

- In Deployer, go to the **Servers > IS & TN** page.
- Click **Add Remote Server Alias**.

Deployer opens the Integration Server Administrator to the **Settings > Remote Servers > Create Alias** page of the Integration Server that hosts Deployer.

- In Integration Server Administrator, define the remote servers by completing the fields in the **Remote Server Alias Properties** area as described in *webMethods Integration Server Administrator's Guide*. You should define the following as remote servers:
 - All target Integration Servers and Trading Networks servers.
 - The Integration Server that hosts Deployer, if you will be using it as a target server (that is, define the Integration Server as a remote server to itself).
- In Deployer, go to the **Servers > IS & TN** and click **Refresh this Page**.

Deployer displays the new server alias to the **Remote Servers List** area of the **Servers > IS & TN** page.

- Select the version of the Integration Server hosting the target runtime from the **Version** box.

For example, if the host Integration Server is running version 9.12, you would select **9.12**. For information about selecting the version, see [“Connecting to Target webMethods Servers” on page 70](#).

- Install the WmDeployerResource package on each Integration Server.

The WmDeployerResource package is the implementation of the operation endpoints for Integration Server and Trading Networks. Install the package as follows:

- a. In Deployer, go to the **Servers > IS & TN** page.

The page lists all Integration Servers you defined as remote servers.

- b. In the **Install** column, select the check box next to each Integration Server on which you want to install the WmDeployerResource package.

- c. Click **Install**.

7. Click **Save**.

Connecting to a Target Optimize Server

Use the following procedure to set up connections to Optimize servers.

➤ To connect to a target Optimize servers

1. In Deployer, go to the **Servers > Optimize** page.
2. For every target Optimize server, click **Configure Optimize Server**. In the **Configure Optimize Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server. Note: You cannot use "localhost" for Optimize servers.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Optimize server. For information about selecting the version, see "Connecting to Target webMethods Servers" on page 70 .
Use SSL	Whether Deployer should use SSL to connect to the server.

3. Click **Configure**. To test the connection, click .

Connecting to a Target Application Platform Server

Application Platform is an add-on for existing products that is installed on Integration Server. To build the user projects, you must first enable the Application Platform Support in a local installation of Integration Server .

➤ To connect to target Application Platform servers

1. In Deployer, go to the **Servers > Application Platform** page.
2. For every source and target Application Platform server, click **Configure Application Platform Server**. In the **Configure Application Platform Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the Application Platform server.
Port	Port for the server. Note: For Integration Server, the port is that of the Integration Server Shared Platform Tomcat instance, rather than the default Integration Server HTTP port. The Shared Platform Tomcat Instance is distinct from the wmTomcat Integration Server Package. It is a separate Tomcat runtime used specifically by the Application Platform add-on.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Application Platform server. For information about selecting the version, see .
Use SSL	Whether Deployer should use SSL to connect to the Application Platform server.

3. Click **Configure**. To test the connection, click .

Connecting to a Target My webMethods Server

Use the following procedure to set up connections to source and target My webMethods Servers.

➤ **To connect to source and target My webMethods Servers**

1. In Deployer, go to the **Servers > MWS** page.
2. For every target My webMethods Server, click **Configure MWS Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the My webMethods Server. For information about selecting the version, see “Connecting to Target webMethods Servers” on page 70.
Root folder aliases	My webMethods Server aliases to use as root folders when selecting pages to deploy. Separate the folders using commas.
Include security dependencies	Whether to include the following in the dependencies list for My webMethods Server assets when creating an MWS deployment set: <ul style="list-style-type: none"> ■ Security realms that contain the assets. ■ User/group/role references in the assets' security ACLs. <p>If the dependencies do not exist on the target My webMethods Servers, include them in the list. You will have to include them in the deployment set. Exclude the dependencies if they do exist on the target My webMethods Servers. For information about resolving dependencies, see “Resolving Dependencies” on page 45.</p>
Maximum Folder Object Count	Maximum number of assets to display within My webMethods Server folders when you are defining and choosing assets to include in an MWS deployment set.
Maximum Folder Depth	Maximum number of My webMethods Server folder levels to display when you are defining and choosing assets to include in an MWS deployment set.

Field	Entry
Enable additional MWS logging	Whether to log debug information about selected assets to source My webMethods Server logs, and assets that Deployer deploys to target My webMethods Server logs.
Exclude Core Task Engine Dependencies	Whether to exclude Task Engine portlets from the dependencies list for task application assets. Exclude the portlets from the list if the target My webMethods Servers host the Task Engine; the portlets are installed with the Task Engine. Include the portlets if the target My webMethods Servers do not host the Task Engine; you will have to include the portlets in the deployment set. For information about dependencies, see “Resolving Dependencies” on page 45 .
Cache Timeout	Length of time queries should remain in the cache unless the cache capacity is exceeded.
Use SSL	Whether Deployer should use SSL to connect to the My webMethods Server. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: You can only use SSL if the My webMethods Server is configured to use SSL. Configure the My webMethods Server's HTTPS port to <i>not</i> request client certificates. For instructions on defining the HTTPS port, see <i>Administering My webMethods Server</i>.</p> </div>

3. Click **Configure**. To test the connection, click .

Connecting to a Target Event Server

Deployer supports deploying assets to Event Servers of version 9.5 or lower.

➤ To connect to target Event Servers

1. In Deployer, go to the **Servers > Event Server** page.
2. Click **Configure Event Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.

Field	Entry
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Event Server. For information about selecting the version, see .
Use SSL	Whether Deployer should use SSL to connect to the Event Server.

Note:
You can only use SSL if the Event Server is configured to use SSL. Configure the server's HTTPS port to *not* request client certificates. For instructions on defining the HTTPS port, see *webMethods Integration Server Administrator's Guide*.

3. Click **Configure**. To test the connection, click .

Connecting to a Target webMethods Broker Server

Important:

webMethods Broker has been deprecated.

For each target Broker, Deployer must connect to the Broker Server that controls that Broker.

> To connect to target Broker Servers

1. In Deployer, go to the **Servers > Broker** page.
2. For every target Broker, click **Configure Broker Server** and complete these fields:

Field	Entry
Name	Name to assign to the Broker Server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: <code>\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "</code>
Host	Host name or IP address of the Broker Server.
Port	Port for the Broker Server.
Version	Version of the Broker Server that matches the version of the project as defined by the host server alias. For information about selecting the version, see "Connecting to Target webMethods Servers" on page 70 .

Field	Entry
Broker Name	Name of the source or target webMethods Broker.
Client Group	Client group for Deployer to use to access the source or target Broker Server. For target Broker Servers, type <code>admin</code> . Note: To connect, Deployer must belong to the specified Broker client group and have access permission.
Context	JNDI context to use when the Broker Server serves as a JNDI provider.
Client Authentication	Whether Deployer should use client authentication to connect to the Broker Server. Select: <ul style="list-style-type: none"> ■ None to connect to the Broker Server without any client authentication. ■ SSL to connect to the Broker Server using SSL authentication. If you select this option, you must complete the Deployer Keystore, Keystore Type, Keystore Password, DeployerTruststore, and Truststore Type fields. Note: You can only use SSL if the Broker Server is configured to use SSL authentication. ■ Basic Authentication to connect to the Broker Server using basic authentication. If you select this option, you must complete the Username and Password fields. Note: You can only use Basic Authentication if the Broker Server is configured to use basic authentication.
Deployer Keystore	Full path to Deployer's keystore file. The keystore contains the SSL credentials (private key and signed certificate) that the Broker Server uses to authenticate Deployer's identity and establish an SSL connection. Required if you specify SSL for Client Authentication .
Keystore Type	File type of Deployer's keystore file. Required if you specify SSL for Client Authentication .
Keystore Password	Password that Deployer uses to access its keystore file. Required if you specify SSL for Client Authentication .
Deployer Truststore	Full path to Deployer's truststore file. The truststore contains the trusted roots for Deployer's SSL certificates. Required if you specify SSL for Client Authentication .
Truststore Type	File type of Deployer's truststore file. Required if you specify SSL for Client Authentication .

Field	Entry
Username	Basic authentication user name. Required for Basic Authentication .
Password	Basic authentication password. Required for Basic Authentication .

3. Click **Configure**. To test the connection, click .

Connecting to a Target Task Engine Server

The following procedure applies when the target Task Engine is running on Integration Server. To connect to a Task Engine instance, running on My webMethods Server, see [“Connecting to a Target My webMethods Server” on page 75](#).

> To connect to target Task Engine servers

1. In Deployer, go to the **Servers >Task Engine** page.
2. For every target Task Engine server, click **Configure Task Engine Server**, and complete the following fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	The password for the user account.
Version	Version of the Task Engine server.
Use SSL	Whether Deployer should use SSL to connect to the server.

3. Click **Configure**. To test the connection, click .


Connecting to a BPM Process Model Server

A BPM process model server is an Integration Server that hosts the webMethods Process Engine and executes business processes.

➤ **To connect to target BPM (ProcessModel) servers**


1. Make sure the Process Audit Log database component is installed and Integration Server is configured to write to it. For instructions, see *Installing Software AG Products*.
2. In Deployer, go to the **Servers > BPM(ProcessModel)** page.
3. For every target ProcessModel server, click **Configure BPM(ProcessModel) Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the server.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the server. For information about selecting the version, see “Connecting to Target webMethods Servers” on page 70 .
Use SSL	Whether Deployer should use SSL to connect to the server.

4. Click **Configure**. To test the connection, click .
5. In Designer, duplicate the logical-to-physical server mapping (defined for each ProcessModel server) on the source and target ProcessModel servers for each model you want to deploy. In the Integration Server Administrator for each of the servers, do the following:
 - a. Define the physical servers in the mapping as remote servers. For instructions, see *webMethods Integration Server Administrator's Guide*.


Note:

Each Integration Server hosting a process model in a cluster must be configured to use the same alias name and port number as the remote alias defined for the cluster. For more information about deploying to clustered Integration Servers, see [“Deploying to Clustered Integration Servers” on page 223](#).

- b. Go to the **Packages > Management** page and click  for the WmDesigner package.

- c. Click **Add Logical Server** and complete these fields:

Field	Entry
Name	Name of a logical server in the mapping for the ProcessModel server. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
	Note: In clusters, both the source and target ProcessModel servers must use the same logical server name.
Physical Server	Physical server to which the logical server is mapped. This should be the same value as the remote server alias name you defined in step 5a.

- d. Click **Add Logical Server**.
- e. Repeat these steps to duplicate the rest of the mapping.
- f. Repeat these steps for every process model you want to deploy.
6. Install the WmDeployerResource package on each ProcessModel server that will run process steps. In Deployer, go to the **Servers > IS & TN** page; the page lists all ProcessModel servers you defined as remote servers. In the **Install** column, select the check box next to each ProcessModel server and click **Install**.
7. If a process model to deploy includes a task, go to the **Packages > Management** page on the model's source and target ProcessModel servers, click  for the WmTaskClient package, and identify the My webMethods Server that hosts the task.

Connecting to a Target AgileApps Cloud Server

Use the following procedure to set up connections to target AgileApps Cloud servers.

> To connect to target AgileApps Cloud servers

1. In Deployer, go to the **Servers > AgileApps Servers** page.
2. Click **Configure AgileApps Server** and complete these fields:

Field	Entry
Name	Name to assign to the server. The name can be up to 32 characters long and cannot contain spaces or the following special characters:

Field	Entry
	\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Server URL	The URL of the AgileApps Cloud server.
User Name	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the AgileApps Cloud server. For information about selecting the version, see .

3. Click **Configure**. To test the connection, click .

If the connection is successful, the following message is displayed: **Successfully connected to AgileApps Server.**

Connecting to a Target Universal Messaging Server

Note that Deployer and the Universal Messaging server must have the same release version. Deployer cannot connect to Universal Messaging servers of versions lower than the Deployer version. For example, with Deployer 10.1 you can connect only to a Universal Messaging 10.1 server, but you cannot connect to a Universal Messaging 9.12 server.

> To connect to a target Universal Messaging server

1. In Deployer, go to **Servers > UniversalMessaging**.
2. Click **Configure UniversalMessaging Server** and specify values in the fields listed in the following table.

Field	Entry
Name	The name to assign to the Universal Messaging server. The upper limit for the length of the name is 32 characters. The name cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Realm URL	The URL of the Universal Messaging server in the following format: <i>protocol://host:port</i> where: ■ valid values for <i>protocol</i> are

Field	Entry
	<ul style="list-style-type: none"> ■ nsp (Universal Messaging Socket Protocol) ■ nhp (Universal Messaging HTTP Protocol) ■ nsps (Universal Messaging Socket Protocol Secure, using SSL/TLS) ■ nhps (Universal Messaging HTTP Protocol Secure, using SSL/TLS) <p>If you use nsps or nhps, you must also specify the details of the Deployer keystore and truststore in the designated fields.</p> <ul style="list-style-type: none"> ■ <i>host</i> - the host on which the server is running. ■ <i>port</i> - the port number of the server.
Version	The version of the Universal Messaging server. Deployer can only connect to a Universal Messaging server of the same release version as Deployer.
Client Authentication	Indicates which type of client authentication to use to connect to the Universal Messaging server: <ul style="list-style-type: none"> ■ None - No client authentication required. ■ SSL - To use SSL, make sure that the Universal Messaging server is configured for SSL authentication, select this option, and specify the details of the Deployer keystore and truststore in the designated fields. ■ Basic Authentication - To use basic authentication, make sure that the Universal Messaging server is configured for basic authentication, select this option, and specify a valid username and password in the designated fields.
Deployer Keystore	Required for the nsps and nhps protocol. The full path to the location of the keystore file.
Keystore Password	Required for the nsps and nhps protocol. The password to access the keystore file.
Deployer Truststore	Required for the nsps and nhps protocol. The full path to the truststore file.
Truststore Password	Required for the nsps and nhps protocol. The password to access the truststore file.
Username	Required for basic authentication. The username of a user account that can access the Universal Messaging server.
Password	Required for basic authentication. The password of the specified user account.

- Click **Configure**. To test the connection, click .

Connecting to a Deployment Endpoint for EDA

Every Software AG installation contains suite-wide EDA assets (event type schema definitions). These assets are common for all OSGi-enabled products running within a particular installation. The Software AG Platform Manager runtime exposes an EDA deployment endpoint that enables users to deploy EDA assets.

Use the following procedure to set up connections to EDA deployment endpoints.

> To connect to target EDA deployment endpoints

- In Deployer, go to the **Servers > EDA** page.
- For every target EDA endpoint, click **Configure EDA Server**. In the **Configure EDA Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the EDA deployment endpoint. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the Software AG Platform Manager runtime.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the Software AG Platform Manager runtime. For information about selecting the version, see “Connecting to My webMethods Server” on page 25 .
Use SSL	Whether Deployer should use SSL to connect to the server. Configure the server's HTTPS port to not request client certificates.

- Click **Configure**. To test the connection, click .

Connecting to a Target Business Rules Integration Server

Use the following procedure to set up connections to target Business Rules Integration Servers.

> To connect to target Business Rules Integration Servers

1. In Deployer, go to the **Servers > Business Rules IS** page.
2. For every and target business rules server, click **Configure Business Rules IS Server**. In the **Configure Business Rules IS Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the runtime. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the Integration Server running business rules.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the business rules runtime. For information about selecting the version, see .
Use SSL	Whether Deployer should use SSL to connect to the server. Configure the server's HTTPS port to not request client certificates. For instructions on defining the HTTPS port, see <i>webMethods Integration Server Administrator's Guide</i> .

3. Click **Configure**. To test the connection, click .

Connecting to a Target Business Rules My webMethods Server

Use the following procedure to set up connections to target Business Rules My webMethods Servers.

> To connect to target Business Rules My webMethods Servers

1. In Deployer, go to the **Servers > Business Rules MWS** page.
2. For every target business rule server, click **Configure Business Rules MWS Server**. In the **Configure Business Rules MWS Server** area, complete the following fields:

Field	Entry
Name	Name to assign to the runtime. The name can be up to 32 characters long and cannot contain spaces or the following special characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Host	Host name or IP address of the My webMethods Server running business rules.
Port	Port for the server.
User	User name for a user account with Administrator authority that Deployer can use to access the server.
Password	Password that is associated with the user name.
Version	Version of the business rules runtime. For information about selecting the version, see .
Root Context	The URL that you use to access My webMethods Server includes the server host name and port number (<code>http://host_name:port_number</code>). However, if necessary, you can also specify a root context path (<code>http://host_name:port_number/root_context</code>). For example, <code>http://localhost:8585/mws</code> .
Use SSL	Whether Deployer should use SSL to connect to the server. Configure the server's HTTPS port to not request client certificates. For instructions on defining the HTTPS port, see <i>webMethods Integration Server Administrator's Guide</i> .

3. Click **Configure**. To test the connection, click .

Creating Repository Deployment Projects

A *repository deployment project* identifies the user-created assets on a repository that you want to deploy to target servers. To create a project, you assign the project a name and set its properties, and then you authorize users to perform the project tasks.



When you create a project, Deployer automatically creates an HTML home page for the project. You can modify this page to contain instructions or notes about the project that you want users to view. For example, you might want to list the target servers for the users who will perform the mapping task, or you might want to provide instructions for users who will test the deployed solution.

For instructions about creating and managing deployment projects, see [“Creating a Project” on page 209](#).

Creating a Deployment Set from Repository Assets

You identify the assets to include in the project using *deployment sets*. Each deployment set identifies the user-created assets you want to deploy from a source repository to a target server. A single project can include deployment sets with assets from different types of webMethods applications. For example, a single project can include Integration Server deployment sets and ProcessModel deployment sets.

> To create a deployment set

1. In Deployer, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. In the right-hand pane, click  **Define**.
5. Click **Create Set**.
6. Specify the following fields:

Box	Entry
Name	Name to use for the deployment set. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Description	Description for the deployment set. The description length has no limit and can include any characters.

7. Click **Create**.

Identifying the Source Repository for a Deployment Set

The source repository is *always* the repository on which the composites are built. If a repository you want to use as a source is not in the list of available repositories, you have not yet created the alias for the repository. Connect to the repository as described in [“Connecting to a Source Repository” on page 69](#) and then click **Refresh this Page** to update the list of repositories.

> To identify the source repository for a deployment set

1. On the **Deployer > Projects > project > Define** page, in the left-hand pane in the **Name** column, click the deployment set from which you will select the composites to deploy.

2. In the **Select** column of the **Select Repository** area, click *repository_alias* that contains composites to add to the deployment set.

You can select only one repository.

3. Click **Save**.


Deployer adds the repository as a child of the deployment set in the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane.


Deployer displays the  icon for the repository.


Selecting Composites

You can add only assets from composites that are included in the source repository. The repository can contain several composites, and you can deploy assets from any composite in the repository, but you cannot add assets from more than one repository to one deployment set. Assets in one composite in the repository can have dependencies on one or more assets in other composites in the repository.

> To select composites for a deployment set



1. In the **Deployment Sets** area of the **Deployer > Projects > project > Define** page, click  for every repository for which you want to add assets.

In the right-hand pane, Deployer displays the  icon for runtime types contained in the repository you selected.

2. Click the plus sign to expand the  runtime types to display the composites they contain.

Deployer displays the  icon for composites.

3. Click every composite whose assets should be part of the deployment set.

To select all of the composites for the runtime type, click . To select all of the composites available on the repository, click .

4. Click **Save**.

Deployer adds the runtime type and the child composites as children of the deployment set in the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane.

Selecting Individual Assets from Composites

For Integration Server, CloudStreams, Trading Networks, and Broker (including JNDI) assets, you can choose to add individual assets from composites rather than the entire composite. Adding


only some of the assets to the deployment set instead of the entire composite is referred to as *partial deployment*.


Note:

For the partial deployment of JNDI assets, export JMS destinations into one JNDI context at a time. Partial deployment of JMS destinations into multiple JNDI contexts is not supported.

Perform the following to select individual assets from composites.

> To select individual assets for a deployment set

1. In the **Deployment Sets** area of the **Deployer > Projects > project > Define** page, click  for every repository for which you want to add assets.

In the right-hand pane, Deployer displays the  icon for runtime types contained in the repository you selected.

2. Click the plus sign to expand the  runtime types to display the composites they contain.

Deployer displays the  icon for composites.

3. Click the name of the composite from which you want to select individual assets.

Deployer displays the composite in the **Select Components** area of the right hand pane.

4. Click the plus sign to expand the  composite to reveal the assets it contains.

5. Click the assets that should be included in the deployment set.

6. Click **Save**.

Deployer displays the  icon for a partial composite with the assets you selected as its children in the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane.

Deploying ACLs through a Deployment Set

Deploying ACLs associated with My webMethods Server Groups

If you want to deploy ACLs that are associated with My webMethods Server groups, you must perform the following tasks.

1. Create an MWS deployment set to deploy the My webMethods Server groups to the target Integration Servers.
2. Create an IS & TN deployment set containing the ACLs associated with the My webMethods Server groups.

3. Mark the unresolved dependencies for the My webMethods Server groups as **Exists**.
4. Deploy the ACLs.

Deploying ACLs associated with LDAP Groups

If you want to deploy ACLs that are associated with LDAP groups, perform the following tasks.



1. Configure the LDAP groups on the target Integration Servers.
2. Create an IS & TN deployment set containing the ACLs associated with the LDAP Groups.
3. Mark the unresolved dependencies for the LDAP groups as **Exists**.
4. Deploy the ACLs.

Resolving Dependencies

Deployer can determine when assets that are in a composite require other assets. The assets that require assets from other composites are called *dependent* assets, while the assets that are required are called *referenced* assets.

Deployer identifies missing referenced assets as follows:

- *Unresolved dependencies* are those dependencies that are present in the repository, but not in the deployment set. Deployer enables you to add, ignore, or automatically resolve those dependencies. Deployer checks dependencies automatically and shows one of the following in the **Unresolved Dependencies** column for the deployment set:

Deployer shows...	When an asset in a composite contains...
	Unresolved dependencies. You can resolve dependencies automatically or manually. <ul style="list-style-type: none"> ■ For information about resolving dependencies automatically, see “Resolving Dependencies Automatically” on page 91. ■ For information about resolving dependencies manually, see “Resolving Dependencies Manually” on page 92.
	No unresolved dependencies.

- *Missing dependencies* are those dependencies that are not available in the repository, so you cannot add them to your deployment set. You can set Deployer to ignore missing dependencies when you create the project (see [“Creating a Project” on page 209](#)) or when you check unresolved dependencies. For more information about setting missing dependency options while checking for unresolved dependencies, see [“Resolving Dependencies Manually” on page 92](#).

Resolving Dependencies Automatically

Perform the following procedure to set Deployer to resolve dependencies automatically.

➤ **To set Deployer to resolve dependencies automatically**

1. In the **Unresolved Dependencies** column for the deployment set, click **Check**.

Deployer displays the **Unresolved Dependencies** page.

2. Click one of the following:

Option	Description
Auto resolve by Composite	<p>Deployer automatically resolves all the unresolved dependencies for the composite in the deployment set. Deployer checks for the dependencies in the repository. If the dependent composites are available in the repository, Deployer adds the composites to the deployment set.</p> <p>If the referenced composites are not available in the repository, Deployer cannot add the composites to the deployment set. You can then choose to ignore the missing dependencies.</p>
Auto resolve by Asset	<p>Deployer automatically resolves the <i>partial</i> addition of composite at the asset level. For example, if AssetA is dependent on AssetB in CompositeB, then Deployer adds only AssetB, instead of the entire composite (CompositeB).</p> <p>If the referenced assets are not available in the repository, Deployer cannot add the composites to the deployment set. You can then choose to ignore the missing dependencies.</p>

Deployer lists the missing dependencies on the **Missing Dependencies** page when you click **Check** in the **Unresolved Dependencies** column. You cannot add missing dependencies to the deployment set.

3. To set Deployer to ignore the missing dependencies perform the following:
 - a. Click **Ignore Missing Dependencies (Project Level)**.
 - b. Click **Apply**.
4. Click **Save**.

Resolving Dependencies Manually

Perform the following procedure to resolve dependencies manually.

➤ **To manually resolve dependencies**



1. In the **Unresolved Dependencies** column for the deployment set, click **Check**.

Deployer shows all unresolved dependencies on the **Unresolved Dependencies** page as follows:

- The **Referenced Asset Composites** column lists the missing referenced assets.
 - The **Unset/Add/Ignore** column offers the possible ways you can resolve the unresolved dependency.
 - The **Assets** column shows the dependent assets.
2. Click one of the following options to set how Deployer should resolve each unresolved dependency. If you want to resolve all assets in a composite the same way, you can set the resolution at the composite level rather than at the level of the individual assets.

Option	Description
Unset	This is the default status. If you click Unset after you have made another selection (Add or Ignore) Deployer resets the assets to an unresolved status. Use this option if you set the assets in a composite to either Add or Ignore and want to start over.
Add	<p>Adds the referenced asset to the deployment set. Use this option if the referenced asset does not exist on the target server and you want to deploy the referenced asset to it.</p> <p>For Integration Server, CloudStreams, Trading Networks, or Broker (including JNDI) assets, you can choose to add the referenced asset or the entire composite that contains the referenced asset. Adding only the referenced asset to the deployment set instead of the entire composite is referred to as <i>partial deployment</i>.</p> <p>Note: For partial deployment of JNDI assets, export JMS destinations into one JNDI context at a time for partial deployment. Partial deployment of JMS destinations into multiple JNDI contexts is not supported.</p>
Ignore	<p>Ignores the asset. Use this option if you want to bypass dependency checking for the referenced asset so you can continue working.</p> <p>Note: Before deploying, make sure that either the referenced asset exists on the target server or that the referenced asset is unnecessary. If the referenced asset does not exist on the target server, Deployer might not deploy correctly; if it can deploy, the deployed assets will not run correctly.</p>

3. Click **Save**.

- Deployer moves dependencies you resolved using the **Add** option to the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane. If you added only a subset of Integration Server, Trading Networks, CloudStreams, or Broker (including JNDI assets) assets rather than an entire composite (partial deployment), Deployer displays  as a sibling of .
- Deployer adds dependencies you resolved using the **Ignore** option to the **Name** column of the **Deployer > Projects > project > Define** page, but the dependency remains listed on the **Unresolved Dependencies** page.

Resolving Conflicts

A *conflict* occurs when a dependent asset is available in multiple composites or when different assets in one composite share the same asset name. Deployer displays conflicts on the same page as unresolved assets.

> To resolve conflicts

1. In the **Unresolved Dependencies** column for the deployment set, click **Check**.

Deployer shows all conflicted assets on the **Unresolved Dependencies** page as follows:

- The **Referenced Conflict Asset Composites** column lists the assets in conflict.
- The **Unset/Add/Ignore** column offers the possible ways you can resolve the conflict.
- The **Assets** column shows the dependent assets.

2. Click one of the following options to set how Deployer should resolve each conflict.

Option	Description
Unset	This is the default status. If you click Unset after you have made another selection (Add or Ignore) Deployer resets the assets to an conflicted status. Use this option if you set the assets in a composite to either Add or Ignore and want to start over. Note: After you use Unset to clear your previous selection, you must set the asset to either Add or Ignore .
Add	Adds the referenced assets to the deployment set. Use this option to deploy the referenced assets.
Ignore	Ignores the conflict. Use this option to ignore the referenced assets.

3. Click **Save**.

- Deployer moves dependencies for which conflicts were resolved using the **Add** option to the **Name** column of the **Deployer > Projects > project > Define** page, in the left-hand pane.
- Deployer ignores the conflicts for those assets which you set to **Ignore**.



Using Deletion Sets

You use *deletion sets* to identify those assets you want to delete from the target server during deployment. When you deploy the project, Deployer deletes the assets defined in the deletion set from the target server and then deploys assets defined for the deployment set from the source repository to the target server. If Deployer does not display a server you want to use as a target, you have not yet set it up to work with Deployer. After you create a deletion set, you should identify the target servers from which to delete assets, add the assets you want to delete in the deletion set, and resolve dependencies.

For repository-based deployment, Deployer supports deletion sets for all runtimes.

Creating a Deletion Set

> To create a deletion set

1. In Deployer, go to the **Deployer > Projects** page.
2. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
3. In the **Name** column, click the project.
4. In the right-hand pane, click  **Define**.
5. Click **Create Set**.

Deployer displays the Create Set properties in the right-hand pane.

6. Complete the following fields:

Box	Entry
Set	Deletion
Name	Name to use for the deletion set. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * ; , + = > < ' ' "
Description	Description for the deletion set. The description length has no limit and can include any characters.

7. Click **Create**.

Deployer displays the deletion set in the left-hand pane in the **Deletion Sets** area.

8. Perform the tasks in [“Identifying Servers for a Repository-Based Deletion Set”](#) on page 96 to identify the servers that contain the assets to add to the deletion set.

Identifying Servers for a Repository-Based Deletion Set

➤ To identify servers for a repository-based project

1. On the **Deployer > Projects > project > Define** page, in the left-hand pane in the **Name** column of the **Deletion Sets Repository** area, click the deletion set for which to identify servers.
2. In the **deletion_set > Select Server** area in the right-hand pane, select the runtime type of the server to include in the deletion set from the **Select Server Type** list.

Deployer displays a list of all servers of the specified type that are set up to work with your system.

3. Select the check box next to each target server that contains assets to add to the deletion set.
4. Click **Add**.

Deployer displays the server you added to **Deletion Sets** area in the left-hand pane.


5. Perform the tasks in [“Adding Assets to a Repository-Based Deletion Set”](#) on page 96 to add assets to the deletion set.

Adding Assets to a Repository-Based Deletion Set

When you map a deletion set to target servers, Deployer identifies assets that depend on the assets you want to delete and lets you resolve those dependencies. However, Deployer can only detect dependencies among assets from the same type of product (for example, among Integration Servers). It cannot detect dependencies among assets from different products (for example, among Integration Servers and ProcessModel servers). Make sure that assets you want to delete for one type of product are not required by assets of other types of product.

When you delete some ActiveTransfer assets, other associated assets might also be deleted. For more information on these asset dependencies, see [“ActiveTransfer Asset Dependencies in Deletion Sets”](#) on page 97.

➤ To add assets to a repository-based deletion set

1. In the **Deletion Sets** area, under the deletion set to which to add assets, click the server () that contains assets you want to add to the deletion set.
2. In the **Select Assets or Asset Components** area in the right-hand pane, expand the tree to display the assets on the servers and select the check box next to each asset to add to the deletion set.
3. Click **Save**.

Deployer displays the assets you selected in the left-hand pane in the **Deletion Sets** area. The assets are grouped by asset categories. For example, Integration Server assets are grouped by isfile, ispackages, and so on.

ActiveTransfer Asset Dependencies in Deletion Sets

Before adding ActiveTransfer assets to deletion sets, consider these dependencies. Deletion of the assets in the deletion sets, also deletes the dependent assets for which Deployer displays a warning.


Asset	Deleting...
Server Instance	An ActiveTransfer Server instance also deletes all ports created for the instance.
Virtual File System	A parent virtual folder also deletes the child virtual folders.
User Template	A user template also deletes all associated users.
Partner Mapping	A partner also deletes all virtual folders associated with the partner.

In deletion sets, when you delete some assets, the asset associations are ignored.

Deleting This Asset...	Does Not Delete...
ActiveTransfer User	Any virtual folders or events associated with the user.
Partner Mapping	Any user or events associated with the partner.

Resolving Dependencies in Repository-Based Deletion Sets

For repository-based projects, Deployer checks dependencies in deletion sets automatically and shows one of the following in the **Unresolved Dependencies** column for the deployment set:

Deployer shows...	When a deletion set contains...
	Unresolved dependencies.

Deployer shows...**When a deletion set contains...****Note:**

You must resolve unresolved dependencies or deployment will fail.

No unresolved dependencies.

Note:

Deployer does not check dependencies in deletion sets for runtime-based projects. For a full explanation of unresolved dependencies for a repository-based project, see [“Resolving Dependencies” on page 91](#).

Perform the following procedure to set Deployer to resolve dependencies for repository-based deletion sets.

> To resolve dependencies for a repository-based deletion set

1. In the **Unresolved Dependencies** column for the deployment set, click  **Check**.

Deployer displays the **Unresolved Asset References** page.

2. Perform one of the following:
 - a. To automatically resolve all unresolved dependencies for the composite, click **Auto resolve all missing references**.
 - b. To manually resolve unresolved dependencies, select the check box in the **Add** column for each referenced asset to add to the deletion set and click **Add**.

5 Product Build Properties and Supported Assets for Repository-Based Deployment

■ About Product Build Properties and Supported Assets	101
■ ActiveTransfer	101
■ AgileApps	106
■ API Gateway	107
■ Application Platform	115
■ BPM Process Development	117
■ Broker	122
■ Business Rules	123
■ Digital Event Services Assets	124
■ Event Routing	125
■ Event Server	125
■ Integration Server	126
■ Mobile Support	189
■ My webMethods Server	190
■ Optimize	194
■ Trading Networks	195
■ Task Engine	198
■ Universal Messaging	198

■ Other Product Assets	200
------------------------------	-----

About Product Build Properties and Supported Assets

The topics included under a product name give details about:

- The product assets supported for repository-based deployment
- Product build properties or other product-specific details required by some of the products when building assets from a source repository into composites

ActiveTransfer

You can deploy the ActiveTransfer assets listed in the following table. The table also specifies the ID, dependencies, and supported substitution variables for each asset type.

Asset	Asset Type ID	Dependencies and Substitution Variables
Server Instance	serverInstance	None
Server Configuration (Ports)	port	<p>Dependencies:</p> <p>Server instance</p> <p>Substitution Variables :</p> <p>Host</p> <p>Port</p> <p>Keystore Location</p> <p>Keystore Password</p> <p>Keystore Private Password</p> <p>RSA Key Location</p> <p>DSA Key Location</p>
Virtual File System VFS		<p>Dependencies:</p> <ul style="list-style-type: none"> ■ User ■ Partner mapping <p>Substitution Variables :</p> <p>VFS URL</p> <p>IS Passive</p> <p>Username</p> <p>Password</p>

Asset	Asset Type ID	Dependencies and Substitution Variables
		ProxyAlias Keystore Location Keystore Password Keystore Key Password SSH Private key SSH Private Key password PGP Public key PGP public key pass
Scheduled Events	scheduleAction	Dependencies: <ul style="list-style-type: none">■ Partner mapping■ Virtual folders Substitution Variables : <p>Is Active</p> <p><i>For copy file action:</i></p> <p>DestinationURL UserName Password ProxyAlias</p> <p><i>For find file action:</i></p> <p>FileURL UserName Password ProxyAlias</p> <p><i>For move file action:</i></p> <p>DestinationURL UserName Password ProxyAlias</p>

Asset	Asset Type ID	Dependencies and Substitution Variables
		<p><i>For rename file action:</i></p> <ul style="list-style-type: none"> NewFileName <p><i>For zip file action:</i></p> <ul style="list-style-type: none"> ZipFilePath UserName Password ProxyAlias <p><i>For unzip file action:</i></p> <ul style="list-style-type: none"> DestinationURL UserName Password ProxyAlias <p><i>For email action:</i></p> <ul style="list-style-type: none"> EmailFrom EmailTo <p><i>For write content to file action:</i></p> <ul style="list-style-type: none"> FilePath <p><i>For encrypt file action:</i></p> <ul style="list-style-type: none"> EncryptionFile <p><i>For decrypt file action:</i></p> <ul style="list-style-type: none"> DecryptionFile DecryptionFilePassword <p><i>For execute script action:</i></p> <ul style="list-style-type: none"> WorkingDirectory
Post-Processing Events	postProcessEvent	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Users ■ Partner mapping ■ Virtual folders

Asset	Asset Type ID	Dependencies and Substitution Variables
		Substitution Variables :
		Is Active
		<i>For copy file action:</i>
		DestinationURL
		UserName
		Password
		ProxyAlias
		<i>For find file action:</i>
		FileURL
		UserName
		Password
		ProxyAlias
		<i>For move file action:</i>
		DestinationURL
		UserName
		Password
		ProxyAlias
		<i>For rename file action:</i>
		NewFileName
		<i>For zip file action:</i>
		ZipFilePath
		UserName
		Password
		ProxyAlias
		<i>For unzip file action:</i>
		DestinationURL
		UserName
		Password

Asset	Asset Type ID	Dependencies and Substitution Variables
		<p>ProxyAlias</p> <p><i>For email action:</i></p> <p>EmailFrom</p> <p>EmailTo</p> <p><i>For write content to file action:</i></p> <p>FilePath</p> <p><i>For encrypt file action:</i></p> <p>EncryptionFile</p> <p><i>For decrypt file action:</i></p> <p>DecryptionFile</p> <p>DecryptionFilePassword</p> <p><i>For execute script action:</i></p> <p>WorkingDirectory</p>
User Template	userTemplate	<p>Dependencies:</p> <p>Server instance</p> <p>Substitution Variables :</p> <p>Default Template</p>
ActiveTransfer Users	user	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Server instance ■ Partner mapping ■ My webMethods user profiles ■ User Template <p>Substitution Variables :</p> <p>Disable login</p> <p>SSH public keys</p> <p>File based encryption file</p> <p>File based decryption file</p> <p>File based decryption key pass</p>

Asset	Asset Type ID	Dependencies and Substitution Variables
Partner Mapping	partnerMapping	Dependencies: Trading Networks partners Substitution Variables : None

Usage Notes

- The values of the substitution variables are case sensitive.
- In the **Asset Properties** section of the Target Substitutions and Source Values screen, the substitution variables for event actions appear in the following format:
 - *If error action:* **ErrorTask.Action Type.User-specified Action Name.Field Name.[Indexed position in the action list]**. For example, ErrorTask.SendEmail.SendAlert.EmailTo.[5].
 - *Other actions:* **Action Type.User-specified Action Name.Field Name**. For example, ExecuteScript.ISscript.WorkingDirectory.
- You migrate My webMethods user profiles when migrating My webMethods, and Trading Networks partner profiles when migrating Trading Networks.
- To track and monitor the deployment of ActiveTransfer assets, make sure that audit logging is enabled in My webMethods for the ActiveTransfer logs.

AgileApps

Deployer version 9.12 and higher supports deploying assets to AgileApps Cloud Servers version 10.7 or higher.

You can export as packages and deploy using Deployer the following AgileApps Cloud assets:

- Application
- Class
- Mapping Profile
- Object
- Package Data Item
- Page
- Report
- Resource
- Site

- Team
- Team Data Sharing
- Translation Workbench
- Web Tab
- Package
- Web and REST Services

The following assets are exported only as dependent assets of the main AgileApps Cloud asset types and cannot be exported as standalone assets:

- Widgets Page
- Document
- Function
- Component
- Class
- Sidebar
- Role
- Mapping Profile
- Business Hours Calendar

When you export an AgileApps Cloud asset that has a dependency on other assets, the dependent assets are also exported with the main asset. For example, when you export an **Application** asset, its dependent assets, such as **Tasks** and **Agent**, also get exported.

API Gateway

You can deploy the following API Gateway assets with webMethods Deployer:

- APIs
- Policy
- Policy actions
- Aliases
- Packages
- Plans
- Administrator Settings
- Assertions

API Gateway does not stage the following API Gateway assets:

- Approval Configurations
- Access Profiles

API Gateway cannot stage the following assets, but you can stage them as Integration Server assets:

- Users
- Groups
- Keystores
- Truststores
- Outbound Proxy
- URL Aliases

When using API Gateway Assets, note the following points:

- APIs, global policies, and packages are maintained in the same state on the target servers as the state they had when built into composites using Asset Build Environment.
- In webMethods Deployer, you cannot redeploy an API that already exists in Active state.
- You cannot delete administrator settings and assertions using a deletion set.
- When deploying partial composites that have hard dependencies, for example policy actions for policies, the dependent assets are added in the deployment set automatically.

API Gateway-Specific Build Properties

When building composites from the API Gateway assets, set the following API Gateway-specific properties in the `master_build/build.properties` file. Do not save clear text password values in the file. Specify the values of the passwords as command line arguments when running the build script.

When a property has a default value, the default is included in the property description.

apigateway.buildLocalRepoOnly

Optional. Indicates whether to create only a local repository, or create an asset repository as per the ABE requirements. The property can be useful in a DevOps scenario to compare assets in a Version Control System (VCS). Valid values:

- `true` - build only a local repository
- `false` - build a repository following the ABE requirements

Default: `false`

build.output.dir

Required. Based on the value of `apigateway.buildLocalRepoOnly`, set this property as follows:

- If `apigateway.buildLocalRepoOnly=true`, specify the directory of the VCS.
- If `apigateway.buildLocalRepoOnly=false`, specify the root directory in which the build script will place the output (composites and descriptors) of the build. This directory path will also get included in the build index.

apigateway.repo.createFromLocalRepo

Optional. Indicates whether the build script will use the location specified in `apigateway.repo.localRepo.path` to create the repository as per the ABE requirements. Valid values:

- `true` - use the location in `apigateway.repo.localRepo.path`. When you set the `createFromLocalRepo` property to `true`, make sure that `apigateway.buildLocalRepoOnly=false`
- `false` - do not use the location in `apigateway.repo.localRepo.path`.

Default: `false`

apigateway.repo.localRepo.path

Required only when `apigateway.repo.createFromLocalRepo=true`. The path to the local working copy of the VCS (or another directory on the local file system), from which the build script will create the repository as per the ABE requirements.

apigateway.assets.file

Required only when `apigateway.repo.createFromLocalRepo=false`. List the assets to export from the source API Gateway and build into composites. Valid values are: ["ASSERTION", "APPLICATION", "API", "ALIAS", "POLICY", "POLICY_ACTION", "PLAN", "PACKAGE", "ADMINISTRATOR_SETTING", "PORTAL_GATEWAY", "SUBSCRIPTION"]. The property points to a JSON file that contains the JSON payload for the specified assets. For samples of JSON payloads that you can use in different scenarios, see [“JSON Payload Samples” on page 110](#).

Default: `*` (exports and builds all assets from the source API Gateway)

apigateway.is.url

Required only when `apigateway.repo.createFromLocalRepo=false`. The URL of the Integration Server that hosts API Gateway.

apigateway.is.username

Required only when `apigateway.repo.createFromLocalRepo=false`. The name of a user account that can access the target API Gateway Server. Make sure that the user you specify here is assigned to an API Gateway group that has import and export assets access privileges.

apigateway.is.password

Required only when `apigateway.repo.createFromLocalRepo=false`. The password of the user account you specified in `apigateway.is.username`.

apigateway.ssl.keystore.file

Required only when communicating with the source API Gateway over SSL. The fully qualified path to the keystore file location.

apigateway.ssl.keystore.type

Required only to communicate over SSL. The type of the keystore.

Default: JKS

apigateway.ssl.keystore.password

Required only to communicate over SSL. The password to use for the specified keystore file.

apigateway.ssl.keyalias

Required only to communicate over SSL. The key alias from the specified keystore file.

apigateway.ssl.keyalias.password

Required only to communicate over SSL. The password of the specified key alias.

apigateway.logging.level

Optional. The logging level of the ABE build script.

Default: OFF

JSON Payload Samples

The following sections include JSON payload samples that you can use for an asset type.

All Asset Types

Export all API Gateway assets:

```
{}
```

API Assets

- Export all APIs and the application.

```
{
  "types": [
    "api"
  ]
}
```

- Export all APIs without the registered application.

```
{
  "types": [
    "api"
  ],
  "includeOptions": {
    "includeApplications": false
  }
}
```

- Export only the APIs that contain API Gateway in their name.

```
{
  "types": [
    "api"
  ],
  "scope": [
    {
      "attributeName": "apiName",
      "keyword": ".*API Gateway.*"
    }
  ],
  "includeOptions": {
    "includeApplications": false
  }
}
```

- Export only the APIs with the specified IDs.

```
{
  "types": [
    "api"
  ],
  "scope": [
    {
      "attributeName": "id",
      "keyword":
"1c36033a-ecb6-41ce-ad66-bada5cebe85e|8d505388-5524-4df5-bd3c-62c4bb5f41d0"
    }
  ],
  "includeOptions": {
    "includeApplications": true
  }
}
```

- Export the APIs with names that start with API Gateway and contain "search" in the description.

```
{
  "types": [
    "api"
  ],
  "scope": [
    {
```

```

        "attributeName": "apiName",
        "keyword": "API Gateway.*"
    },
    {
        "attributeName": "apiDescription",
        "keyword": ".*search.*"
    }
],
"condition": "and",
"includeOptions": {
    "includeApplications": false
}
}

```

Policy Assets

- Export all policies, including the global policy and threat protection rules.

```

{
  "types": [
    "policy"
  ]
}

```

- Export only global policies.

```

{
  "types": [
    "policy"
  ],
  "scope": [
    {
      "attributeName": "policyScope",
      "keyword": "GLOBAL"
    }
  ]
}

```

- Export the threat protection rules.

```

{
  "types": [
    "policy"
  ],
  "scope": [
    {
      "attributeName": "policyEnforcements.stageKey",
      "keyword": "threatProtection"
    }
  ]
}

```

- Export all policy actions, including the global denial of service and denial of service by IP threat protection rules.

```

{
  "types": [
    "policy_action"
  ]
}

```



```
]
}
```

- Export all threat protection rules and policies.

```
{
  "types": [
    "policy_action",
    "policy"
  ],
  "scope": [
    {
      "attributeName": "policyEnforcements.stageKey",
      "keyword": "threatProtection"
    },
    {
      "attributeName": "id",
      "keyword": "globalipdos|ipdos"
    }
  ],
  "condition": "or"
}
```

Alias Assets

Export all aliases.

```
{
  "types": [
    "alias"
  ]
}
```

Application Assets

- Export all applications.

```
{
  "types": [
    "application"
  ]
}
```

- Export all applications without including the associated APIs.

```
{
  "types": [
    "application"
  ],
  "includeOptions": {
    "includeApis": false
  }
}
```

Assertion Assets

Export all assertions.

```
{
  "types": [
    "assertion"
  ]
}
```

Plan Assets

Export all plans.

```
{
  "types": [
    "plan"
  ]
}
```

Package Assets

- Export all packages and all dependent assets associated with the packages, such as:

- dependent plans

- associated APIs

- registered applications

- subscriptions

```
{
  "types": [
    "package"
  ]
}
```

- Export all packages with the subscriptions, but without the applications registered for the associated APIs.

```
{
  "types": [
    "package"
  ],
  "includeOptions": {
    "includeApis": false,
    "includeApplications": false,
    "includeSubscriptions": true
  }
}
```

Subscription Assets

Export all subscriptions with the dependent packages, plans, and APIs.

```
{
  "types": [
    "subscription"
  ]
}
```

Administrator Setting Assets

Export all administrator settings.

```
{
  "types": [
    "administrator_setting"
  ]
}
```

Portal Gateway Assets

Export the portal gateway destination.

```
{
  "types": [
    "portal_gateway"
  ]
}
```

Application Platform

You can deploy the following types of Application Platform assets using Deployer.

- Bundle - An OSGi bundle.
- Service - An OSGi service.
- WebApp - A web application.
- Config - An OSGi dynamic configuration because these assets represent an OSGi-specific item unlike the web application.
- JndiResource - A JNDI resource injection. This asset defines how to inject an OSGi dynamic configuration into the JNDI space of a web application. This asset does not represent the configuration itself.

Application Platform composites can contain the following assets:

- bundle composite: Bundle, WebApp, Service
- war composite: WebApp
- properties composite: Config, JndiResource

The following table lists the assets that you can export, the asset type ID for each asset, and the asset dependencies.

Asset	Asset Type ID	Dependencies, Substitutions, and Other Considerations
OSGi Bundle	Bundle	Each "bundle" composite always contains exactly one Bundle asset. The Bundle asset can depend on zero or more Service assets that are almost always provided by other "bundle" composites.
OSGi Service	Service	Each "bundle" composite can contain zero or more Service assets. The Service asset does not have any dependencies on other assets.
OSGi Dynamic Configuration	Config	Each "properties" composite always contains exactly one Config asset. The Config asset does not have any dependencies on other assets. By default, all properties in the "properties" file represented by the "properties" composite are translated to substitutions for the Config asset.
JNDI resource injection	JndiResource	<p>When a "properties" composite defines a JNDI resource that needs to be injected into a web application, it will contain exactly one JndiResource asset in addition to the main Config asset.</p> <p>When the JndiResource asset defines an injection of an OSGi Service into the web application, it will have a dependency on the respective Service asset. JndiResource assets that do not define a Service injection can never have dependencies on other assets. When a "properties" composite contains a JndiResource, all properties of the respective properties file except those that describe the JNDI name and target web application context, are translated to substitutions for the Config asset of the "properties" composite.</p>
Web Application	WebApp	The "war" composite always contains exactly one WebApp asset. The "bundle" composite can contain a single WebApp asset if the composite represents a "web bundle" (WAB). The WebApp asset can have dependencies on one or more JndiResource assets - one for every JNDI resource described in the web.xml.

BPM Process Development

You can prepare BPM Process Development process models (.process files) created with Software AG Designer for deployment with Deployer. The build process enables you to filter the deployable assets by process ID, process version, or a combination of the two.

The following table describes the asset type that you can export, with its ID and dependencies.

Asset	Asset Type ID	Description
Process model	bpmprocess	A typical business process model contains references to and dependencies on a number of other runtime assets. For successful deployment, each of these other assets must be checked in to your version control repository. Before you attempt to deploy a business process model, ensure that you have all process elements and dependent assets checked into your repository, including:

Note: Deployer cannot determine dependencies for dynamically referenced processes. You should manually deploy such dependencies.

- The entire process project, including the project directory, the build.xml file, and all .process files.
- The generated process package in Integration Server.
- Any Integration Server packages that contain Integration Server assets (such as IS documents or services) that are invoked from the process.
- Any Composite Application Framework (CAF) assets (especially tasks), any rule assets, Trading Networks assets, referenced process assets from other process projects, or other assets that are invoked from the process.

Building BPM Assets into Composites

You can configure and run a build script that creates a composite and ACDL descriptor for specific process models. Deployer uses the resulting composite and descriptor files to deploy the process development assets as part of a deployment project.

Note that you get different results when you generate a process model manually in Software AG Designer and when you deploy a process model with Deployer.

Deploying Generation Receipts

Deployer does not deploy generation receipts when deploying Designer process models. The primary reason for this is that the generation receipt stores a logical-to-physical server mapping and the only information available to Deployer is in the composite file, which does not contain any mapping information.

When you regenerate a process, the regeneration process uses the mapping information to determine whether a logical server mapping has changed from the previous generation. If the mapping has changed, the regeneration process connects to the previous logical server and cleans up the process-related assets there (for example, deleting associated triggers and services), in addition to creating the new assets on the new logical server. With no generation receipt, this mapping information is not available and the clean-up procedure cannot occur.

Because repository-based deployment does not generate a generation receipt, when you deploy a process to a target environment using repository-based deployment, you cannot then redeploy that process using that target environment as a source for a runtime-based deployment project. In runtime-based deployment Deployer requires the generation receipt to determine dependencies. If you want to redeploy the process, you must first rebuild the process in the target environment.

Note:

If a user connects Designer to a target environment and regenerates a deployed process to servers other than those deployed to the original process, the wrapper services and triggers on the original servers are not deleted. Wrapper services and triggers are still generated on the new logical servers, and existing services and triggers on unchanged logical servers remain unchanged. You must perform a manual cleanup on old logical servers. If you do not, multiple triggers and services could run in the process.

If you regenerate a previously deployed process to the same logical servers, Designer creates a new generation receipt so that future regeneration processes are handled correctly.

Deploying Process Images

When you deploy a process model, the associated process model image and icon images are not deployed with the model. As a result, any webMethods Monitor APIs that use the process or icon images will not perform as expected with the deployed process.

Approaches for Building Composites from BPM Process Development Assets

To create a composite and descriptor file for each .process file to be deployed, you can use one of the following approaches:

- For a standard use in a defined deployment environment, configure the build properties in the build.properties file as described in [“Setting Build Properties” on page 58](#). Run the master-level build script in the \bin directory of your installed Asset Build Environment environment as described in [“Running the Build Script” on page 65](#).
- Use the project-level build script, in which the build script is run in a process project directory. Running this build script is useful because process project directories can be located anywhere in a user’s file system (that is, outside the deployment environment). This approach enables

you to create the composite and descriptor files in any process project directory, regardless of location. After the composites and descriptors are built, they can be moved or copied into a deployment repository and indexed by the build script in order to make them available to Deployer. Project-level build script behavior is governed by the build.xml file in the process project directory. For information about configuring the build.xml file, see [“Configuring the Process Project build.xml File” on page 120](#).

Be aware that in either mode, the build script processes *only* those process projects that contain a build.xml file. Process projects without a build.xml file are ignored, even if a process model is explicitly named.

Software AG Designer adds a build.xml file to a process project automatically for projects created with Process Development version 8.2 SP1 and later. If a process project has no build.xml file, you can copy the build.xml file from any other project. If the build.xml was not modified in the originating project, you do not have to modify the build.xml file to copy it from one project to another.

BPM-Specific Build Properties

You must set the following BPM-specific build properties in the build.properties file when running the build script in the master_build directory.

bpm.acdl.model.ids

Optional. Semicolon-separated list of process IDs (that is, a concatenation of process project name and process model file name).

If the process does not exist in the source directory or one of its subdirectories, the value is ignored. If left empty, all process model IDs are included. This property can be used with or without the bpm.acdl.model.version property.

bpm.acdl.bam.model.ids

Optional. Semicolon-separated list of BAM process model IDs to add to the composite. BAM process models are tracking only and do not declare any dependencies. For example: testProcessProject/testProcessModel;testProcessProject2/testProcessModel2. If the process does not exist in the source directory or a child directory of it, the value is ignored. If left empty, no process model IDs are included. All process models are assumed to be BPM processes and will generate the standard process dependencies. This property can be used with or without the bpm.acdl.model.ids or bpm.acdl.model.version properties.

bpm.acdl.model.version

Optional. Version number of the process models to include in the build. Enter a single integer value (example, 1).

If this value is set, the build script includes only those process models in the source directory or one of its subdirectories that match the version number. If left empty, the build script includes

process models of all versions. You can use this property with or without the `bpm.acdl.model.ids` property.

Configuring the Process Project `build.xml` File

Note that a BPM `build.xml` file must exist in each process project, but you only configure this file when the build script is run in the process project directory. Do not configure the `build.xml` file if the build script is run in the `master_build` directory. The property settings in `build.xml` are used *only* when the build script is running in the process project directory. If the build script is running in the `master_build` directory, the `build.xml` properties are overridden by the properties in the `build.properties` file.

Note:

A BPM `build.xml` file is included by default if the Designer process project is created in Designer 8.2 or later. If the process project is imported from previous versions of Designer, you must add it manually to the process project folder.

When the script is run in the process project directory, the build script assumes that the source directory and the output directory are the same as the current directory.

➤ To configure the BPM `build.xml` file

1. In the process project you want to work with, open the `build.xml` file in a text editor.
2. Define the following properties as needed:

Property	Definition
<code>default.sag.install.dir=path/to/directory</code>	Required. The location of the Software AG suite installation directory. The specified directory must contain the <code>common/lib</code> directory.
<code>sag.master.build.dir=path/to/directory</code>	Required. The location of the Deployer environment you installed with Software AG Installer. The specified directory must contain the <code>master_build</code> directory.
<code>default.bpm.acdl.model.ids=processID1;processID2</code>	Optional. A semicolon-separated list of process IDs (that is, a concatenation of process project name and process model file name). For example: <code>testProcessProject/testProcessModel;testProcessProject2/testProcessModel2</code> If the process does not exist in the source directory or a child directory of it, the build script ignores the value. If left empty, all process model IDs are included. This property can be used with or without the <code>default.bpm.acdl.model.version</code> property.

Property	Definition
<code>default.bpm.acdl.model.version=n</code>	Optional. A single integer value that is matched to the version number of process models in the source directory or a child directory of it. The build script includes only models with a version number equal to this value in the build. If left empty, the build script includes all process model versions. You can use this property with or without the <code>model.ids</code> property.
<code>default.bpm.acdl.bam.model.ids</code>	<p>Optional. A semicolon-separated list of BAM process model IDs. For example: <code>testProcessProject/testProcessModel;testProcessProject2/testProcessModel2</code></p> <p>You must specify all BAM process models because they are for tracking purposes only and do not need to declare any dependencies. If the process does not exist in the source directory or a child directory of it, the build script ignores the property.</p> <p>If left empty, all process models are assumed to be BPM processes. The build script treats any process models it builds but that are not included in this list as BPM processes and generates the standard process dependencies for them. You can use this property with or without the <code>bpm.acdl.model.ids</code> or <code>bpm.acdl.model.version</code> properties.</p>

3. Save the file.

Running the Build Script in a Process Project Directory

Use this procedure if you want to run the build script on a single process project. In this case, the build script creates composite and descriptor files within the process project directory. The build script considers all process files in the process project directory and its sub-directories, as specified by the properties in `build.xml`.

➤ To run the build script in a process project directory

1. Define any of the available `build.xml` properties as described in [“Configuring the Process Project `build.xml` File”](#) on page 120.
2. Open a command prompt window.
3. Change to the process project directory where you want to run the build script.

4. Run this command: `ant`

Note:

Ensure that you have modified your system variables to contain the path name of the build script. Otherwise, you must type the full path name.

The build script processes the specified process model files and creates corresponding composite and descriptor files in the current directory. After the composite and descriptor files are built, you can move or copy the files to a deployment repository and index them for use with Deployer.

5. If you want to use the composite and descriptor files for use with Deployer, perform the following additional steps:
 - a. Move or copy the files to the BPM subdirectory of the repository.
 - b. Run the following command from the *Software AG_directory* \common\AssetBuildEnvironment\bin directory:

For this platform...	Run the following command...
Windows	<code>build.bat createIndex</code>
UNIX	<code>build.sh createIndex</code>

The `createIndex` script indexes the files you moved or copied to the repository.

Broker

Important: webMethods Broker has been deprecated.

Using Deployer, you can deploy your Broker assets and JNDI assets to another Broker, and share the assets that you created in Broker with other applications.

webMethods Broker supports exporting the following assets to Deployer:

- Broker assets such as clients, client groups, and document types.
- JNDI assets such as JMS queues and JMS topics created by webMethods JNDI providers.

Some Broker assets have dependencies on other Broker components. When you export assets that have dependencies on other assets, the corresponding dependent assets are also exported. For example, when you export a client group, the document types belonging to that client group are also exported.

The following table lists the assets that you can export, the asset type ID for each asset and its dependencies.

Asset	Asset Type ID	Dependencies
Client group	ClientGroup	Document Type
Client	Client	Client Group
Document type	DocumentType	None
JMS destination (JMS queues and JMS topics created by webMethods JNDI providers)	JMSDestination	None

Business Rules

The following table describes the user-created webMethods Business Rules assets that you can export, the asset type ID for each asset and its dependencies.

Asset	Asset Type ID	Dependencies						
Action	ruleaction	<p>Actions have dependencies based on the type of action as follows:</p> <p>For this type of action... The dependency is...</p> <table border="0"> <tr> <td>Service</td> <td>Integration Server service</td> </tr> <tr> <td>Process</td> <td>BPM process</td> </tr> <tr> <td>New Data</td> <td>Business rules data model</td> </tr> </table>	Service	Integration Server service	Process	BPM process	New Data	Business rules data model
Service	Integration Server service							
Process	BPM process							
New Data	Business rules data model							
Data model	ruledatamodel	Optionally dependent on a nested data model.						
Decision table	ruledecisiontable	<p><i>Always</i> depends on at least one data model.</p> <p>Optionally dependent on one or more action.</p>						
Event rule	ruleeventrule	<p><i>Always</i> dependent on at least one data model.</p> <p>Optionally dependent on one or more action.</p>						
Rule sets	ruleset	<p><i>Always</i> dependent on one of the following:</p> <ul style="list-style-type: none"> ■ Decision table ■ Event rule 						
Rule project	ruleproject	<ul style="list-style-type: none"> ■ com.softwareag.rules.hotdeploy.project 						

Asset	Asset Type ID	Dependencies
		<p>When you deploy on My webMethods Server, hot deploy the rule project to pre-configured Integration Server connection(s). Valid values are <code>true</code> and <code>false</code>. By default, the value is <code>false</code>.</p> <ul style="list-style-type: none">■ <code>com.softwareag.rules.merge.project</code> <p>When you deploy on My webMethods Server, merge the rule project with the existing rule project that has the same name in the repository. Valid values are <code>true</code> and <code>false</code>. By default, the value is <code>false</code></p> <p>For more information, see <i>Working with Business Rules in My webMethods</i>.</p>

Business Rules-Specific Build Properties

When building composites from business rules assets, set the following Business Rules-specific property in the `master_build/build.properties` file.

RULES.skip.on.validation.warning

Specifies whether the build script should create a rule project archive and composite when the rule project issues validation warnings. Set to:

- `true` to create a project archive and composite.
- `false` to turn off project archive and composite creation. This is the default.

Digital Event Services Assets

The following table lists the assets you can export and deploy to Digital Event Services (DES) deployment endpoints, as well as the asset type ID and dependencies:

Asset	Asset Type ID	Dependencies
DES Annotation	<code>DesAnnotation</code>	None.
DES Event Type	<code>DesEventType</code>	Optionally dependent on assets of type <code>DES Annotation</code> and <code>DES Event Type</code> , and dependent on one asset of type <code>DES Event Type Information</code> .
DES Event Type Information	<code>DesEventTypeInfo</code>	None.

Event Routing

The following table lists the asset you can export and deploy to Event Routing deployment endpoints, as well as the asset type ID and dependencies:

Asset	Asset Type ID	Dependencies
Event Type schema definitions	XSD	None.

Event Server

Note: Deployer supports deploying assets only to Event Servers of version 9.5 or lower.

You can deploy the following Event Server assets:

- Continuous query projects
- Event type projects

The following table lists the dependencies and properties that you can substitute when using Deployer to deploy assets to an Event Server.

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations								
Continuous query projects	XML	<p>Dependencies:</p> <p>None.</p> <p>Substitution values:</p> <p>For each database source defined in the project, you can substitute the following properties:</p> <table border="1"> <thead> <tr> <th>Property</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>JdbcConnectionUrl</td> <td>The URL to be used to connect to the database.</td> </tr> <tr> <td>Username</td> <td>The user name to be used to connect to the database.</td> </tr> <tr> <td>Password</td> <td>The password for the user specified in Username.</td> </tr> </tbody> </table>	Property	Description	JdbcConnectionUrl	The URL to be used to connect to the database.	Username	The user name to be used to connect to the database.	Password	The password for the user specified in Username .
Property	Description									
JdbcConnectionUrl	The URL to be used to connect to the database.									
Username	The user name to be used to connect to the database.									
Password	The password for the user specified in Username .									

A database source in the composite file has the same URL and user name values that it did at design time. These values become the default values for the **JdbcConnectionUrl** and **Username** properties at deployment time. If you leave the

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		<p>JdbcConnectionUrl and Username property fields empty at deployment time, the default values (those that were assigned to the database source at design-time) are assigned to the database source when it is deployed. The password for a database source, however, is never included in the composite file. Therefore, the Password property does not have an associated default value that can be used for deployment. Because it has no default value, you must explicitly assign a value to the Password property for each database source in the composite file.</p>

Note:

If multiple database sources use the same **Password** value, select all of them in the **Configurable Components** panel, specify the password value and then click **Save Substitutions**. Doing this will assign the specified password to all of the selected data sources in one step.

Integration Server

The assets that you can export from Integration Server and deploy using Deployer belong to the following asset group types.

- Integration Server administrative assets such as ACLs, extended settings, groups, JMS aliases, JNDI aliases, ports, scheduled tasks and users.

Note:

Integration Server does not generate any assets for the Native Users, ACLs, or Groups. This includes the following:

- **Native Users.** Administrator, Default, Replicator, and Developer.
 - **Native Groups.** Everybody, Administrators, Anonymous, Developers, and Replicators.
 - **Native ACLs.** Administrators, Anonymous, Replicators, Developers, Default, and Internal.
- CloudStreams configuration assets such as database, OAuth token, streaming provider, and subscriber option settings.
 - Integration Server packages.
 - Adapter runtime assets, used by all WmART based adapters, .NET asset used by webMethods Microsoft Package, and CloudStreams assets used by webMethods CloudStreams.

Integration Server Administrative Assets

This section describes the following:

- [Adding Administrative Assets to the Source Directory](#)
- [Global Values for Integration Server Administrative Assets](#)
- [Integration Server Administrative Assets and Substitution Values](#)

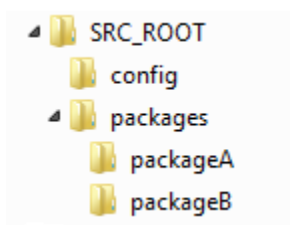
Adding Administrative Assets to the Source Directory

To include administrative assets in the composite for deployment, you must manually copy or check in the *Integration Server_directory* \instances*instance_name*\config folder to the source directory.

Note:

To include CloudStreams administrative assets in the composite for deployment, you must manually copy or check in the *WmCloudStreams\config* folder to the source directory, at the same location as the user-defined packages.

Software AG recommends that you structure your source directory to contain all of the administrative assets to be included in the composite as shown in the following example:



In this example, you would define SRC_ROOT as the value of the build.source.dir property in the build.properties file. For more information about the build.properties file, see [“Setting Build Properties” on page 58](#).

If the asset you want to include in the composite contains passwords or encrypted values, you must also add the passman.cnf and txnPassStore.dat files in the SRC_ROOT/config directory.

When run, the build script creates the following:

- A composite named *package_name.zip* for each package included in the source directory, where *package_name* is the name of the package (with a composite type ID of ispackage). For this example, the files would be named packageA.zip and packageB.zip.
- A composite named isconfiguration.zip (with a composite type ID of isconfiguration) that contains the administrative assets contained in the config directory.

For more information about building composites for repository-based deployment, see [“Building Composites” on page 57](#).

The following table lists the files and directories that you must manually copy or check in to the source directory in order to build Integration Server administrative assets for deployment.

Asset	Files to copy or check into the source directory
ACLs	<ul style="list-style-type: none"> ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\users.cnf ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\acls.cnf
Broker settings	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\dispatch.cnf
Cache manager	All Ehcache configuration files located in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\Caching directory.
	<p>Note: Do not include system cache managers. For example, do not include cache managers whose names start with “SoftwareAG”.</p>
Certificate settings	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\server.cnf
Client certificate	The Asset Build Environment extracts this asset from the database.
	<ul style="list-style-type: none"> ■ If you are using the embedded database, the Asset Build Environment must have access to <i>Integration Server_directory</i> \instances\<i>instance_name</i>\db\embedded directory. ■ If you are using an external database, the Asset Build Environment requires access to the JDBC configuration files.
CSRF guard configuration	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\security\csrf\csrfguard.cnf
	<p>Note: Integration Server creates this file only when the CSRF guard option is enabled in Integration Server Administrator. For more information, see <i>webMethods Integration Server Administrator’s Guide</i>.</p>
Enhanced parser	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\parsing.cnf
Enterprise Gateway configuration	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\security\enterprisegateway\enterpriseGatewayRules.cnf
Extended settings	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\server.cnf
Global variables	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\globalVariables.cnf
Groups	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\users.cnf
IS Packages	For all the assets contained in the IS package ACDL file, you must check in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\ <i>package_name</i> directory (where <i>package_name</i> is the package for which the ACDL is required) to the source directory.

Asset	Files to copy or check into the source directory
	<p>The following files are required to retain ACL information for the different assets in a package:</p> <ul style="list-style-type: none"> ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\acfs.cnf ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\aclmap_sm.cnf ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\acllist.cnf ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\aclread.cnf ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\aclwrite.cnf
JDBC driver alias	<p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\jdbc\driver*.xml</p> <p>Note: The Asset Build Environment does not extract default driver aliases that are installed with Integration Server.</p>
JDBC functional alias	<p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\jdbc\function*.xml</p>
JDBC pool alias	<p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\jdbc\pool*.xml</p> <p>Note: The Asset Build Environment does not extract the embedded database pool alias.</p>
JMS aliases	<p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\jms.cnf</p>
JNDI aliases	<p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\jndi\jndi_*.properties</p>
Keystore alias	<p>The Asset Build Environment extracts keystore alias details from the *_config.xml files stored in the <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\security\keystore directory and then reads the actual keystore binary file (the .jks or .p12) from the value found in the "location" field of the keystore alias definition.</p>
LDAP configuration	<p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\ldap.cnf</p>
Metadata	<p><i>Integration Server_directory</i> \instances\<i>instance_name</i>\packages\WmAssetPublisher\config\assetpublisher.cnf</p>
Mobile Support configuration	<ul style="list-style-type: none"> ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\packages\WmMobileSupport\config\mobileSyncComponents.cnf

Asset	Files to copy or check into the source directory
	<ul style="list-style-type: none"> ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\packages\WmMobileSupport\config\mobileApp.cnf <p>For more information about these assets, see “Mobile Support” on page 189.</p>
MQTT connection alias	<i>Integration Server_directory</i> \ instances\ <i>instance_name</i> \packages\ <i>package_name</i> \config\commonmessaging.cnf
Ports	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \packages\ <i>package_name</i> \config\listeners.cnf
Proxy server alias	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \config\proxy.cnf
Proxy server bypass	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \config\server.cnf
	<p>Note: The Asset Build Environment does not extract this file if <code>watt.net.proxySkipList</code> is set to <code>localhost</code>. For more information about <code>watt.net.proxySkipList</code>, see <i>webMethods Integration Server Administrator’s Guide</i>.</p>
Quiesce mode configuration	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \config\quiesce.cnf
Reliable messaging configuration	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \config\reliableMessaging.cnf
Remote server alias	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \config\remote.cnf
	<p>Note: The Asset Build Environment does not extract remote server aliases named “local”.</p>
SAML token issuer	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \config\security\saml\trusted_saml_issuers.cnf
Scheduled tasks	<p>The Asset Build Environment extracts this asset from a database, and requires either of the following:</p> <ul style="list-style-type: none"> ■ If you are using the embedded database, the Asset Build Environment must have access to <i>Integration Server_directory</i> \<i>instances</i>\<i>instance_name</i>\db\embedded directory. ■ If you are using an external database, the Asset Build Environment requires access to the JDBC configuration files.
SFTP server alias	<i>Integration Server_directory</i> \ <i>instances</i> \ <i>instance_name</i> \config\sftp\sftpServerAliases.cnf

Asset	Files to copy or check into the source directory
SFTP user alias	<ul style="list-style-type: none"> ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\sftp\sftpUserAliases.cnf ■ <i>Integration Server_directory</i> \instances\<i>instance_name</i>\config\sftp\identities directory and its contents
Truststore alias	The Asset Build Environment extracts truststore alias details from the *_config.xml files stored in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\security\keystore directory and reads the actual truststore binary file (.jks or .p12) from the value found in the "location" field of the truststore alias definition.
Universal Messaging connection alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\messaging.cnf
URL alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\ <i>package_name</i> \config\urlalias.cnf
Users	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\users.cnf
Integration Cloud Accounts	<i>Integration Server_directory</i> \config\integrationlive\connections.cnf
Integration Cloud Applications	<i>Integration Server_directory</i> \config\integrationlive\applications directory
Integration Cloud Settings	<i>Integration Server_directory</i> \config\integrationlive\accounts.cnf
Web service endpoint alias	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\endpoints*.cnf
Web service policy	All .policy files in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \config\wss\policies directory
CloudStreams OAuth Tokens	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \WmCloudStreams\config\auth\ds-oauthDetails.xml
CloudStreams Streaming Providers	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \WmCloudStreams\config\resources\streaming\config.xml
CloudStreams Streaming Subscribers	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \WmCloudStreams\config\resources\streaming\config.xml
CloudStreams Administration Database	WmCloudStreams\config\resources\wst-config.properties

Asset	Files to copy or check into the source directory
CloudStreams Large Data Configuration	WmCloudStreams\config\resources\wst-config.properties

Integration Server-Specific Build Properties

When building composites from Integration Server assets, set the following Integration Server-specific property in the `master_build/build.properties` file:

is.acdl.config.dir

Specifies the full path of the Integration Server config directory. For example:

Software AG_directory \IntegrationServer\config

If you set the `build.source.project.dir` property, you must also specify a value for this property.

If you set this property, you must also specify a value for `build.source.project.dir`. If you do not specify a value for `build.source.project.dir`, the build script ignores this value.

Global Values for Integration Server Administrative Assets

The following table lists the global administrative assets and substitution values for each that Integration Server supports for deployment. The assets presented in the table are deployed as assets within the `isconfiguration` composite.

Asset	Asset Type ID	Substitution Values
Cache manager	iscachemanager	<p>Reload Cache Managers After Deployment (<code>reloadCacheManagersAfterDeployment</code>)</p> <p>Specifies whether the cache managers on the target servers are started after deployment.</p> <ul style="list-style-type: none"> ■ True starts the cache manager on the target servers after deployment. If the cache manager is already in the start state on the target Integration Server, a value of True will restart the cache manager with the new configuration. ■ False does not start the cache manager on the target servers after deployment. If the cache manager is already in the start state on the target Integration Server, a value of False will not restart the cache manager.

Asset	Asset Type ID	Substitution Values
		<ul style="list-style-type: none"> ■ None uses the value specified at the cache manager level Reload property. If the Reload property of the cache manager has the value None, cache manager is not started after deployment. <p>If the cache manager is in a start state on the target Integration Server, a value of None will not restart the cache manager. If the cache manager is in a shutdown state on target Integration Server, the cache manager will not be started.</p> <p>The default is None.</p>
<p>Note: Integration Server deploys a cache manager that uses BigMemory or Terracotta Server Array only if you have the appropriate Terracotta and Integration Server licenses. For more information about licenses, see <i>webMethods Integration Server Administrator's Guide</i>.</p>		
Ports	isport	<p>Enable Ports After Deployment (enablePortsAfterDeploy)</p> <p>Specifies whether the ports on the target servers are enabled after deployment.</p> <ul style="list-style-type: none"> ■ True enables the port on the target servers after deployment. ■ False disables the port on the target servers after deployment. ■ None uses the value specified at the port level Enable property. The default is None.
Scheduled task	istask	<p>Suspend tasks during deployment (suspendTasksDuringDeploy)</p> <p>Specifies whether you want to suspend all scheduled tasks during deployment.</p> <ul style="list-style-type: none"> ■ True indicates that you want to suspend all tasks during deployment. ■ False indicates that you want all tasks to run as scheduled during deployment. ■ None indicates that you want use the value specified by the Suspend During Deployment value of each scheduled task asset. If the task has

Asset	Asset Type ID	Substitution Values
		the value of None , no action is taken during deployment. The default value is None .
		<p>Activate tasks after deployment (activateTasksAfterDeploy)</p> <p>Specifies whether you want to activate all scheduled tasks after deployment.</p> <ul style="list-style-type: none"> ■ True indicates that you want all tasks to activate after deployment. ■ False indicates that you want all tasks to suspend after deployment. ■ None indicates that you want use the value specified by the Activate After Deployment value from each scheduled task asset. If the task has the value of None, no action will be taken during deployment.

Integration Server Administrative Assets and Substitution Values

The following sections describe the Integration Server administrative assets and their corresponding substitution values.

ACLs

The following table lists the asset type ID and substitution values for ACL assets.

Asset Type ID	isacl
Substitution Values	None.

Broker Settings

The following table lists the asset type ID and substitution values for Broker settings assets.

Asset Type ID	isbrokersettings
Substitution Values	
Broker Configuration Settings:	

Broker Host	Name (<i>DNSname:port</i> or <i>ipaddress:port</i>) of the machine on which the Broker Server resides.
brokerHost	
Broker Name	Name of the Broker to which the Integration Server connects.
brokerName	
Client Group	Client group to which the Integration Server belongs.
clientGroup	
Client Prefix	A string that identifies the Integration Server to the Broker.
CLIENTPREFIX	
Client Authentication Settings:	
Username	User name required to connect to the Broker.
brokerUser	
Password	Password required to connect to the Broker.
brokerPassword	
Keystore	The full path to the keystore file for the target Integration Server.
certFile	
Keystore Password	Password required to access the SSL certificate in the Integration Server's keystore file.
password	
Use Source Keystore	Specifies whether the target Integration Server uses the keystore file from the source Integration Server.
useSource BrokerKeystore	<ul style="list-style-type: none"> ■ True indicates that the target Integration Server uses the keystore file from the source Integration Server. This is the default. ■ False indicates that the target Integration Server will use a different keystore file than the source Integration Server.
	<p>Note: This configuration value does not correspond to a field or property in Integration Server.</p>
Encryption Settings:	
Truststore	Full path to the Integration Server's client truststore file.
truststore	

Use Source Truststore Specifies whether the target Integration Server uses the truststore file from the source Integration Server.

- useSource Broker Truststore
- **True** indicates that the target Integration Server uses the truststore file from the source Integration Server. This is the default.
 - **False** indicates that the target Integration Server will use a different truststore file than the source Integration Server.

Note:

This configuration value does not correspond to a field or property in Integration Server.

Cache Manager**Note:**

Integration Server deploys a cache manager that uses BigMemory or Terracotta Server Array only if you have the appropriate Terracotta and Integration Server licenses. For more information about licenses, see *webMethods Integration Server Administrator's Guide*.

Asset Type ID

iscachemanager

Substitution Values

cacheManagerName

Name. The name of the cache manager.**Note:**

If a cache manager with the same name exists in both the source and target servers and if the cache manager name is modified while deploying to the target server, the target server will contain both the cache managers, one with the old name as well as the one with the name used while deploying.

urls

Terracotta Server Array URLs. (Optional.)

Comma-separated list of host names and port numbers, one for each server in the Terracotta Server Array. You can add multiple URLs by using this format:
host1:port,host2:port...

Note:

You must specify the Terracotta Server Array URLs only if you have distributed caches in the cache manager.

reload

Reload. Specifies whether the cache manager on the target servers is started after deployment.

- **True** starts the cache manager on the target servers after deployment.
- **False** does not start the cache manager on the target servers after deployment.
- **None** indicates that you want to use the global value specified by the deployment options on the **Deployment Map Properties > Configure Builds by Assets** screen and by the **Reload Cache Managers After Deployment** value in the project map file.

If the cache manager is in a start state on the target Integration Server, a value of **None** will not restart the cache manager. If the cache manager is in a shutdown state on target Integration Server, the cache manager will not be started.

The default is **None**.

Certificate Settings

Asset Type ID iscertificates

Substitution Values

None.

Client Certificates

Asset Type ID isclientcerts

Substitution Values

Certificate Path The name of the file containing the certificate that you want to import. You may specify the file name using an absolute path or using a path that is relative to *Integration Server_directory*. The file must contain an X.509 certificate in DER file format.

certificate Path

CloudStreams Streaming Subscribers

Asset Type ID	iscloudstreamsstreamingsubscribers
Substitution Values	
Provider Name	One of the configured streaming providers.
streaming.sub.prov.name	
Channel Endpoint	The topic/channel name or the endpoint address for the subscription request.
streaming.sub.channel.endpoint	
HTTP Content Type	The HTTP Content-Type header value. Applicable only if the Streaming Providers Client Type field is set to HTTP at source.
streaming.sub.channel.content.type	
HTTP Method	The HTTP method to use for the streaming request made to the service provider. Applicable only if the Streaming Providers Client Type field is set to HTTP at source.
streaming.sub.channel.method	
HTTP Request Body	The optional request entity or message contents to send to the service endpoint. Applicable only for the POST, PUT, OPTIONS, and PATCH HTTP methods and if the Streaming Providers Client Type field is set to HTTP at source.
streaming.sub.channel.entity	
\${Header Name} Header Value	Value for the HTTP header \${Header Name} to be sent to the service endpoint. Applicable only if the Streaming Providers Client Type field is set to HTTP at source.
\${Header Name}	
ESB Service Name	Integration Server service that will be invoked when a streaming notification is received from the provider. Available only if Destination Type is selected as ESB Service at source.
streaming.sub.dest.esb.runas	
Run ESB Service as User	The user name, Integration Server should use when running the ESB

<code>streaming.sub.dest.esb.service</code>	Service. Available only if Destination Type is selected as ESB Service at source.
Journal Log Level	The Integration Server journal log to which the streaming subscription response contents will be logged when a notification is received from the provider. Available only if Destination Type is selected as Journal Log at source.
<code>streaming.sub.dest.log.level</code>	

CloudStreams Streaming Providers

Asset Type ID	<code>iscloudstreamsstreamingproviders</code>
Substitution Values	
Client Type	The type of client CloudStreams will use to provide the streaming functionality. The supported types are as follows:
<code>streaming.prov.client.type</code>	<ul style="list-style-type: none"> ■ Comet - This is a client implementation that uses the Bayeux HTTP protocol (using the CometD library) to communicate with the streaming service. ■ HTTP - An HTTP client type is used for HTTP-based streaming service providers that use a long-lived HTTP connection to send data periodically to the client connected to them.
API Version	The version of the streaming API supported by the provider service.
<code>streaming.prov.api.version</code>	
API Endpoint	The endpoint of the streaming provider.
<code>streaming.prov.endpoint</code>	
Connection Timeout	The streaming connection timeout value in milliseconds.
<code>streaming.prov.conn.timeout</code>	
Read Timeout	The maximum time in milliseconds before which a data packet must be read
<code>streaming.prov.conn.read.timeout</code>	

from the endpoint before a timeout occurs.

SSL Truststore Alias

streaming.prov.ssl.ts.alias

The Integration Server truststore alias to use if the endpoint is SSL based and the provider certificate must be validated as part of the SSL handshake.

SSL Keystore Alias

streaming.prov.ssl.ks.alias

The Integration Server keystore alias to use if the endpoint is SSL-based and the CloudStreams certificate must be provided to the server endpoint as part of the SSL handshake.

Validate SSL Certificate

streaming.prov.ssl.validateCerts

Configuration to enable validation of the provider's certificate for the SSL handshake.

- **true** - Enables validation of the provider's certificate.
- **false** - Disables validation of the provider's certificate.

Basic Authentication Username

streaming.prov.authn.basic.username

The username for HTTP Basic authentication. This is available only if **Authentication Type** is configured as **Basic** at source.

Basic Authentication Password

streaming.prov.authn.basic.pwd

The password for HTTP Basic authentication. This is available only if **Authentication Type** is configured as **Basic** at source.

OAuth Alias

streaming.prov.authn.alias

The OAuth 1.0a or OAuth 2.0 alias. This is available only if **Authentication Type** is configured as OAuth 1.0 or OAuth 2.0 at source.

Authentication ESB Service

streaming.prov.authn.esb.service

User-defined ESB service which provides value for the Authorization header. This is available only if **Authentication Type** is configured as ESB Callback at source.

CloudStreams OAuth Tokens

Asset Type ID

iscloudstreamsoauthtokens

Substitution Values

Consumer ID clientId	The key issued by the service provider and is used by the consumer to identify itself to the service provider.
Consumer Secret clientSecret	The secret key used by the consumer to establish ownership of the consumer key.
Access Token accessToken	The token used for authentication instead of user credentials. This token is issued by the Authorization Server.
Access Token Secret accessTokenSecret	A secret used by the consumer to establish ownership of a given access token. This is available only for OAuth 1.0a configured at source.
Instance URL instanceUrl	Runtime host, for back ends like Salesforce. This is available only for OAuth 2.0 configured at source.
Refresh Token refreshToken	A token used by the client to obtain a new access token without having to involve the resource owner. This is available only for OAuth 2.0 if Refresh Access Token is enabled at source.
Refresh URL refreshUrl	Provider specific URL to refresh an access token. This is available only for OAuth 2.0 if Refresh Access Token is enabled at source.
Custom Refresh Service refreshRequestTypeCustomService	User implemented service for refreshing the access token. This is available only for OAuth 2.0 if Refresh Access Token is enabled and Refresh URL Request is configured to Custom ESB Service at source.

CloudStreams Administration Database Configuration

Asset Type ID	iscloudstreamsadminDATABASECONFIG
Substitution Values	
Database Publishing pg.jdbc.active	Options to control publishing of metrics and events to the database. <ul style="list-style-type: none"> ■ true - Enables publishing. ■ false - Disables publishing.
Publish Performance Metrics pg.PgMenConfiguration.perfDataEnabled	Options to control publishing of key performance indicator (KPI) metrics to the database.

	<ul style="list-style-type: none"> ■ true - Enables publishing of KPI metrics. ■ false - Disables publishing of KPI metrics.
Publish Interval <code>pg.PgMenConfiguration.reportInterval</code>	The interval in minutes at which CloudStreams should publish performance metrics to the database.
Publish Events <code>pg.publish.events</code>	Options to control publishing of error, lifecycle, and policy violation events to the database. <ul style="list-style-type: none"> ■ true - Enables publishing of events. ■ false - Disables publishing of events.

CloudStreams Administration General - Large Data Configuration

Asset Type ID	<code>iscloudstreamslargefileconfig</code>
Substitution Values	
Handle Binary Streams <code>wst.largedata.binaryStreamEnabled</code>	Large data configuration enables CloudStreams to send and receive large binary streams over HTTP/HTTPS. If you enable Handle Binary Streams , then during outbound and inbound invocations, if the stream is greater than the Threshold Size (bytes) , the entire stream is not stored in memory. <ul style="list-style-type: none"> ■ true - Enables handling large data. ■ false - Disables handling large data.
Threshold Size (bytes) <code>wst.largedata.threshold</code>	Large data configuration enables CloudStreams to send and receive large binary streams over HTTP/HTTPS. If you enable Handle Binary Streams , then during outbound and inbound invocations, if the stream is greater than the Threshold Size (bytes) , the entire

stream is not stored in memory. The default is 5242880 bytes.

CSRF Guard Configuration

Asset Type ID `iscsrfguardconfig`

Substitution Values

Enabled	Specifies whether CSRF guard is enabled in Integration Server.
<code>isEnabled</code>	<ul style="list-style-type: none"> ■ True specifies that CSRF guard is enabled. ■ False specifies that CSRF is not enabled. This is the default.
Excluded User Agents	A list of user agents for which Integration Server is not to apply CSRF guard. If CSRF guard is enabled, Integration Server requires that HTTP requests coming from user agents that are not specified as excluded must contain CSRF secure tokens.
<code>excludedUserAgents</code>	
Landing Pages	A list of landing pages for the packages in your Integration Server. A landing page is the home page for a package. Integration Server does not check for CSRF secure tokens in the landing pages, but inserts a token for that page. Integration Server guards all further requests from these landing pages with CSRF secure tokens.
<code>landingPages</code>	
Unprotected URLs	The URLs for which Integration Server is not to check for CSRF secure tokens. If CSRF guard is enabled, Integration Server requires that the requests coming from all URLs that are not specified as unprotected must contain CSRF secure tokens.
<code>unprotectedURLs</code>	
Denial Action	Action that you want Integration Server to perform when it detects that a request does not contain a CSRF secure token or contains an invalid CSRF secure token.
<code>denialAction</code>	<ul style="list-style-type: none"> ■ Error specifies that you want Integration Server to throw an access denied error and terminate the request. This is the default. ■ Redirect specifies that Integration Server is to redirect the user to a confirmation page or the home page of Integration Server Administrator.

Email Notification

Asset Type ID `isemailnotification`

Substitution Values

SMTP Server Server name or IP address of the SMTP server to use to send the messages.

smtpServer

Port Port that Integration Server connects to on the specified SMTP server.

smtpPort

Transport Layer Security Whether any transport layer security is to be used when communicating with the SMTP port. Possible values are: none, explicit, implicit

transportSecurity

Truststore Alias Alias for the truststore that contains the list of certificates that Integration Server uses to validate the trust relationship.

trustStoreAlias

Internal Email Email address to use for critical log entries.

internalEmail

Service Email Email address to use for service failures.

serviceMail

Default Email Charset The default character set.

defaultCharset

Username The username for the account that Integration Server uses to connect to the SMTP server.

userName

Password Password for the account.

password

Extended Settings

Asset Type ID isproperty

Substitution Values

Server configuration parameters that are set as visible. Allows users to specify values for watt properties that are set as visible in the source Integration Server.

File Access Control Configuration

Asset Type ID isfileaccesscontrol

Substitution Values

Allowed Read Paths Semicolon-delimited list of directories to which the services in the pub.file folder in the WmPublic package have read permission.

allowedRead Paths

Allowed Write Paths Semicolon-delimited list of directories to which the services in the pub.file folder in the WmPublic package have write permission.

allowedWrite Paths

Allowed Delete Paths Semicolon-delimited list of directories that the services in the pub.file folder in the WmPublic package have permission to delete.

allowedDelete Paths

Note:

After making changes and deploying fileAccessControl.cnf, you must reload the WmPublic package or restart Integration Server for the changes to take effect.

Global Variables

Asset Type ID isglobalvariable

Substitution Values

Value Values for global variables.

Value

Groups

Asset Type ID isgroup

Substitution Values

None.

Integration Cloud Accounts

Asset Type ID iswmccloudaccount

Substitution Values

Stage The Integration Cloud stage from which the on-premise Integration Server listens for messages. This value refers to the stage created by the user on Integration Cloud.

stage

Note:

Regardless of what the stages are named on Integration Cloud, the stages are deployed using the following names:

- stage00stage01stage02stage99

Allowed On-Premise Hosts Comma-separated list of the on-premise Integration Servers that can listen for messages for a particular account.

allowedOn
PremiseHosts

Integration Cloud Applications

Asset Type ID iswmcloudapplication

Substitution Values

None.

Integration Cloud Settings

Asset Type ID iswmcloudsettings

Substitution Values

User Name The user name the on-premise Integration Server must use to access Integration Cloud.

user

Password The password the on-premise Integration Server must use to access Integration Cloud.

password

Integration Cloud URL The URL of the Integration Cloud server.

wmCloudURL

JDBC Driver Alias

Asset Type ID isjdbcdriveralias

Substitution Values

None.

JDBC Pool Alias Configuration

Asset Type ID isjdbcpoolalias

Substitution Values

Database URL The URL to use to connect to the database.

databaseURL

User ID Database user for the target Integration Server to use to communicate with the database.

userID

Password Password for the specified database user.

password

JDBC Functional Alias

Asset Type ID isjdbcfunctionalalias

Substitution Values

None.

JMS Connection Alias

Asset Type ID isjmsalias

Substitution Values

General Settings:

Connection Client ID The JMS client identifier associated with the connections established by this JMS connection alias.

clientID

User User name required to acquire a connection from the connection factory.

user

Password Password required to acquire a connection from the connection factory.

password

JNDI Connection Protocol Settings:

JNDI Provider Alias Name Alias name of the JNDI provider.

jndi_jndiAlias Name	
JNDI Connection Factory Lookup Name	Lookup name that the connection factory uses to create a connection to the JNDI provider.
jndi_connection FactoryLookup Name	
Automatically Create Administered Objects	Whether Integration Server creates administered objects in the JNDI namespace and on Universal Messaging if the object is not found when Integration Server looks up the object. This applies only when Universal Messaging is the JMS provider and JNDI provider.
jndi_automaticallyCreateAdministeredObjects	<ul style="list-style-type: none"> ■ True indicates that Integration Server creates a destination or connection factory when a JNDI lookup for the object fails. ■ False indicates that Integration Server does not create missing administered objects. This is the default.
Native webMethods Connection Protocol Settings:	
Broker Host nwm_brokerHost	Name (<i>DNSname:port</i> or <i>ipaddress:port</i>) of the machine on which the Broker Server resides.
Broker Name nwm_brokerName	Name of the Broker as defined on the Broker Server.
Client Group nwm_client Group	Name of the client group that you want the target Integration Server to use when it acts as a JMS client.
Broker List nwm_brokerList	Comma delimited list of Broker Servers on which the connection between the target Integration Server (acting as the JMS client) and the webMethods Broker (acting as a JMS provider) can exist.
Keystore nwm_keystore	The full path to the target Integration Server's keystore file.
Truststore nwm_truststore	Full path to target Integration Server client's truststore file.

JNDI Alias

Asset Type ID isjndialias

Substitution Values

Provider URL providerURL	The primary URL of the initial context for sessions with the JNDI provider. The URL specifies the JNDI directory in which the JNDI provider stores JMS administered objects.
Provider URL Failover List providerURL FailoverList	A list of URLs to which the target Integration Server can connect if the connection to the primary JNDI provider becomes unavailable. Separate the URLs with an ampersand, new line, carriage return, or horizontal tab.
Security Principal security Principal	The principal name, or user name supplied by the target Integration Server to the JNDI provider, if the provider requires one for accessing the JNDI directory. For information about whether or not the JNDI provider requires security principal information, consult the product documentation for the JNDI provider.
Security Credentials security Credentials	The credentials, or password, that the target Integration Server provides to the JNDI provider, if the provider requires security credentials to access the JNDI directory. For information about whether or not the JNDI provider requires security credentials, consult the product documentation for the JNDI provider.

Kerberos Settings

Asset Type ID	iskerberosconfig
Substitution Values	
Realm realm	The domain name of the Kerberos server, in all uppercase letters. All the computers managed by the Key Distribution Center (KDC) and secondary KDCs, if any, constitute the realm.
Key Distribution Center Host kdc	The host name of the machine on which the KDC resides.
Kerberos Configuration File conf	The location of the Kerberos configuration file that contains the Kerberos configuration information, including the locations of KDCs, defaults for the realm and for Kerberos applications, and the hostnames and Kerberos realms mappings.
Use Subject Credentials	Whether Integration Server requires a Kerberos V5 Generic Security

useSubjectCredsOnly

Services (GSS) mechanism to obtain the necessary credentials from an existing subject set up by the JAAS authentication module. Subject represents the user or service being authenticated in the JAAS login context.

Keystore Alias

Asset Type ID iskeystorealias

Substitution Values

Location The full path to the keystore file for the target Integration Server.

ksLocation

Password Password associated with this alias that is used to protect the contents of the keystore.

ksPassword

Key Alias Password Password for each key alias residing in the keystore.

keyAlias.
keyAliasName

Use Source Keystore Specifies whether the target Integration Server uses the keystore file from the source Integration Server.

- useSource Keystore
- **True** indicates that the target Integration Server uses the keystore file from the source Integration Server. This is the default.
 - **False** indicates that the target Integration Server will use a different keystore file than the source Integration Server.

Note:
This configuration value does not correspond to a field or property in Integration Server.

LDAP Configuration

Asset Type ID isldapdirectory

Substitution Values

Directory URL The complete URL of the LDAP server. The URL has the format *protocol://hostname:portnumber/DistinguishedName*, where:

`directoryURL` ■ *protocol* is `ldap` for standard connections or `ldaps` for secure connections
host is the host name or IP address of the LDAP server

portnumber is the port on which the server is running. The port is optional. If omitted, the port defaults to 389 for LDAP, or 636 for LDAPS.

DistinguishedName is optional, and is in the form of an LDAP distinguished name (DN), for example "`dc=webMethods, dc=com`", or "`o=webMethods.com`", depending on how your directory is set up.

The default value for `directoryURL` is **`ldap://localhost:389\`**.

Principal <code>principal</code>	The user ID the Integration Server should supply to connect to the LDAP server, for example, <code>o=webm.com</code> or <code>dc=webm,dc=com</code> .
Credentials <code>credentials</code>	The password the Integration Server should supply to connect to the LDAP server, that is, the Principal's password.

Metadata

Note:

Deployer extracts the metadata asset from the `WmAssetPublisher` package. Make sure this package resides in the source folder before extracting the asset.

Asset Type ID `ismetadata`

Substitution Values

CentraSiteURL <code>centraSiteURL</code>	The CentraSite URL to which to publish metadata about Integration Server assets. For example: <code>http://localhost:port/CentraSite/CentraSite</code>
User Name <code>centraSiteUserName</code>	The name of the user account on CentraSite that will be used for publishing and retracting metadata.
Password <code>centraSitePassword</code>	Password associated with User Name .

MQTT Connection Alias

Asset Type ID `ismqttalias`

Substitution Values

Connection Protocol Settings:

Host URL containing the protocol, domain, and port of the MQTT server to which connections created using the MQTT connection alias will connect.

host

Connection Client ID Client identifier for the connections associated with this MQTT connection alias.

clientId

Security Settings:

User User name required for client authentication if basic authentication is required to connect to the MQTT server.

user

Password Password required for the specified user name.

password

Enhanced Parser

Asset Type ID `isparsing`

Substitution Values

Default Partition Bytes Specifies the size, measured in bytes, of the partitions on the heap where the enhanced XML parser stores parsed document information. Specify a suffix of “k” to indicate kilobytes or “m” to indicate megabytes. For example, 10k or 10m.

default
PartitionBytes

Note:

This value is used only when the enhanced parser is invoked and no partition size is specified.

Use Cache Indicates if caching used with the enhanced XML parser.

useCache

- **True** indicates that during parsing of an XML document, Integration Server moves partitions to an off-heap area on disk when the on-heap partition space becomes full and retrieves the partitions as needed during processing.
 - **False** indicates that Integration Server does not cache partitions when parsing an XML document.
-

Maximum Heap Bytes Maximum amount of heap space that the parser can allocate to process documents concurrently. Specify a suffix of “k” to indicate kilobytes, “m” to

<code>maximumHeap Bytes</code>	indicate megabytes, “g” to indicate gigabytes, and “%” to indicate a percentage of the heap space. For example, 10k, 10m, 10g, or 10%.
Maximum Document Bytes <code>maximumDoc Bytes</code>	Maximum amount of heap space that the parser can allocate to process a single document. Specify a suffix of “k” to indicate kilobytes, “m” to indicate megabytes, “g” to indicate gigabytes, and “%” to indicate a percentage of the heap space. For example, 10k, 10m, 10g, or 10%.
UseBigMemory <code>useBigMemory</code>	Indicates if BigMemory is used with the enhanced XML parser. <ul style="list-style-type: none"> ■ True indicates that when parsing an XML document using the enhanced XML parser Integration Server moves partitions to BigMemory once the on-heap cache is full. When BigMemory becomes full partitions are move to disk. Integration Server retrieves partitions from disk or BigMemory as needed. ■ False indicates that when parsing an XML document using the enhanced XML parser Integration Server does not use BigMemory to store cached partitions.
MaximumBig MemoryBytes <code>maximumBig MemoryBytes</code>	Maximum amount of non-heap memory, measured in bytes, to allocate to BigMemory. Specify a suffix of “k” to indicate kilobytes, “m” to indicate megabytes, or “g” to indicate gigabytes. For example, 10k, 10m, or 10g.

Ports

Asset Type ID `isport`

Substitution Values

General values for all ports (email, file polling, FTP, FTPS, HTTP, and HTTPS):

Note:
Deployer extracts the Ports asset from the package in which the port is configured. Make sure this package resides in the source folder before extracting the asset.

Package Name <code>pkg</code>	The package name you want to associate with the port on the target servers. If the port is used as the quiesce port, the port alias should be associated with either the WmRoot or WmPublic package.
Enable <code>enable</code>	Specifies whether the port on the target servers are enabled or disabled after deployment. <ul style="list-style-type: none"> ■ True enables the port on the target servers after deployment. ■ False indicates that you do not want the port enabled on the target servers after deployment.

- **None** indicates that you want to use the global value specified by the deployment options on the **Deployment Map Properties > Configure Builds by Assets** screen and by the **Enable Ports After Deployment** value in the project map file.

Hosts hostList	A comma delimited list that specifies the hosts allowed or not allowed to connect to the target servers through this port.
	<p>Note: If the Access Mode is Global, the host list is ignored. If the Access Mode is Allow, the host list represents the hosts that are denied access to the port. If the Access Mode is Deny, the host list represents the hosts that are allowed access to the port.</p>
IP access type hostAccessMode	<p>Specify the access type of host that is allowed to connect to the target servers through this port.</p> <ul style="list-style-type: none"> ■ Allow indicates that you want Integration Server to allow access by default and to deny the hosts specified in the hosts list. ■ Deny indicates that you want Integration Server to deny access by default and allow only hosts specified in the host list. ■ Global indicates that you want to use the global access settings specified on Integration Server. <p>For detailed information about IP access types and controlling IP access, see <i>webMethods Integration Server Administrator's Guide</i>.</p>
Email port:	
Host Name host	The name of the machine on which the POP3 or IMAP server runs.
Port server_port	(Optional.) The port on the e-mail server to which the target servers can connect.
User Name user	A user name that identifies the target servers to the e-mail server.
Password password	The password associated with the user name that identifies the target servers to the e-mail server.
Run services as user runUser	<p>The user name you want the target servers to use when running the service.</p> <p>The target servers run the service as if the user you specify is the authenticated user that invoked the service. If the service is governed by an ACL, be sure to specify a user that is allowed to invoke the service.</p>

File polling port:

Monitor Directory The directory on the target servers that you want to monitor for files.

monitorDir

Working Directory (Optional.) The directory on the target servers to which the servers should move files for processing after they have been identified in the Monitoring Directory. Files must meet age and file name requirements before being moved to the Working Directory.

workDir

Completion Directory (Optional.) The directory on the target servers to which you want files moved when processing is completed in the Monitoring Directory or Working Directory.

completionDir

Error Directory The directory on the target servers to which you want files moved when processing fails.

errorDir

Enable Clustering Specifies whether the target servers should allow clustering in the Monitoring Directory.

clusterEnabled

- **Yes** indicates that you want the target servers to allow clustering in the Monitoring Directory.
 - **No** indicates that you do not want the target servers to allow clustering in the Monitoring Directory.
-

Run Services as User The user name you want the target servers to use when running the service.

runUser

The target servers run the service as if the user you specify is the authenticated user that invoked the service. If the service is governed by an ACL, be sure to specify a user that is allowed to invoke the service.

Directories are NFS mounted file system For use on a UNIX system where the monitoring directory, working directory, completion directory, and/or error directory are network drives mounted on the local file system.

NFSDirectories

- **No** indicates that you want the listener to call the Java `File.renameTo()` method and move the files from the monitoring directory to the working directory, and from the working directory to the completion and/or error directory. This is the default.
 - **Yes** indicates that you want the listener to first call the Java `File.renameTo()` method to move the files from the monitoring directory. If this method fails, the listener will copy the files from the monitoring directory to the working directory and delete the files from the monitoring directory. This operation will fail if either the copy action or the delete action fails. The same behavior applies when moving files from the working directory to the completion and/or error directory.
-

FTP port:

Port The port number you want to use for the FTP port on the target servers. Select a number that is not already in use on the target servers.

port

Bind address (Optional.) IP address to which to bind this port. Specify the substitute bind address if the target servers have multiple IP addresses and you want the port to use this specific address.

bindAddress

FTPS port:

Port The port number you want to use for the FTPS port on the target servers. Select a number that is not already in use on the target servers.

port

Bind address (Optional.) IP address to which to bind this port. Specify the substitute bind address if the target servers have multiple IP addresses and you want the port to use this specific address.

bindAddress

Secure Clients Only Specify whether you want to prevent the FTPS listener from operating with non-secure clients.

secureclients

- **True** prevents the FTPS listener from operating with non-secure clients.
 - **False** allows the FTPS listener to operate with non-secure clients.
-

HTTP port and HTTPS port:

Port The port number you want to use for the HTTP port or HTTPS port on the target servers. Select a number that is not already in use on the target servers.

port

Bind address (Optional.) IP address to which to bind this port. Specify the substitute bind address if the target servers have multiple IP addresses and you want the port to use this specific address.

bindAddress

Quiesce Settings

Asset Type ID `isquiescesettings`

Substitution Values

Use the following values for the quiesce port on the target servers:

Port Alias The Integration Server port to use as the quiesce port on the target servers.

portAlias

Ensure that the port alias is associated with either the WmRoot or the WmPublic package and is of type HTTP or HTTPS.

System Tasks Disabled Specifies whether the system tasks on the target servers are enabled after deployment.

systemTasksDisabled

	<ul style="list-style-type: none"> ■ True disables the system tasks. ■ False (Default) enables the system tasks.
Alerting Disabled alertingDisabled	Specifies whether alerts or notifications will get triggered on the target servers after deployment <ul style="list-style-type: none"> ■ True disables alerts or notifications. ■ False (Default) enables alerts or notifications.
Auditing Disabled auditingDisabled	Specifies whether audit logging is enabled on the target servers through this port. <ul style="list-style-type: none"> ■ True disables audit logging. ■ False (Default) enables audit logging.
Statistics Data Collector Disabled statisticsDataCollectorDisabled	Specifies whether the statistics data collector is enabled on the target servers after deployment. <ul style="list-style-type: none"> ■ True disables the statistics data collector. ■ False (Default) enables the statistics data collector .

Proxy server alias

Asset Type ID isproxyserveralias

Substitution Values

Host Name or IP Address The host name or IP address of the proxy server.

host

Port Number The port on which this proxy server listens for requests.

port

User Name The user name Integration Server must use when accessing this proxy server.

username

Password The password Integration Server must use to access this proxy server.

password

Proxy Server Bypass

Asset Type ID isproxyserverbypass

Substitution Values

Addresses The fully qualified host and domain name of each server to which you want the Integration Server to issue requests directly.

addresses

Type the host name and the domain name exactly as they appear in the URLs the server uses. To enter multiple names, separate each with commas. You can use the asterisk (*) to identify several servers with similar names. The asterisk matches any number of characters. For example, if you want to bypass requests made to localhost, www.yahoo.com, home.microsoft.com, and all hosts whose names begin with NYC, you would type:

```
localhost,www.yahoo.com,home.microsoft.com, NYC*.*
```

Reliable Messaging Configuration

Asset Type ID isreliablemessaging

Substitution Values

Retransmission Interval The time interval (in milliseconds) for which a reliable messaging source waits for an acknowledgement from the reliable messaging destination before the source retransmits the SOAP message.

retransmission
Interval

Acknowledgement Interval The time interval (in milliseconds) for which the reliable messaging destination waits before sending an acknowledgement for a message sequence. Messages of the same sequence received within the specified acknowledgement interval are acknowledged in one batch. If there are no other messages to be sent to the acknowledgement endpoint within the time specified as the acknowledgement interval, the acknowledgement is sent as a stand alone message.

acknowledgement
Interval

Exponential Backoff Whether to use the exponential backoff algorithm to adjust the retransmission interval of unacknowledged messages. Adjusting the time interval between retransmission attempts ensures that a reliable messaging destination does not get flooded with a large number of retransmitted messages.

exponential
Backoff

- **True** specifies that the successive retransmission intervals must be increased exponentially, based on the specified retransmission interval. For example, if the specified retransmission interval is 2 seconds, and the exponential backoff value is set to true, successive retransmission intervals will be 2, 4, 8, 16, 32, and so on if messages continue to be unacknowledged.
 - **False** specifies that the retransmission interval is not to be adjusted.
-

Inactivity Timeout <code>inactivity Timeout</code>	The length of time for which a reliable messaging source waits for an acknowledgement from a reliable messaging destination before the source stops retransmitting the SOAP message.
	If the reliable messaging source does not receive an acknowledgement within the inactivity timeout specified, it marks the sequence as timed out. You cannot use a sequence if it is timed out. To indicate that there is no inactivity timeout limit, set the value of Inactivity Timeout as -1.
Inactivity Timeout Interval <code>inactivity TimeoutMeasure</code>	The unit of measure for the <code>inactivity Timeout</code> property. You can specify the unit of measurement as seconds, minutes, hours, or days. The default is 60 seconds.
Sequence Removal Timeout <code>sequenceRemoval Timeout</code>	The length of time for which a reliable messaging source waits for an acknowledgement from a reliable messaging destination before it terminates the sequence and removes the sequence from the memory.
Sequence Removal Timeout Interval Measure <code>sequenceRemoval TimeoutMeasure</code>	The unit of measure for the <code>sequenceRemoval Timeout</code> property. You can specify the unit of measurement as seconds, minutes, hours or days. The default is 60 seconds.
In-Order Delivery Assurance <code>invokeInOrder</code>	Whether the messages in a sequence must be delivered to a reliable messaging destination in the same order in which they have been sent by the reliable messaging source. <ul style="list-style-type: none"> ■ True specifies that the messages in a sequence must be delivered to the destination in the same order in which they have been sent. The order in which the messages are sent is indicated by the sequence key of each message. This is the default. ■ False specifies that the delivery of messages in the same order in which they have been sent is not to be enforced.
Maximum Retransmission Count <code>maximum Retransmission Count</code>	The number of times a reliable messaging source must retransmit a message if an acknowledgement is not received from the reliable messaging destination. The value must be an integer within the range of 1 and 256 (inclusive). The default is 10. To indicate that there is no limit to the number of retransmission attempts, set the value of Maximum Retransmission Count as -1.
Storage Type <code>storageType</code>	Specifies whether Integration Server uses the persistent or non-persistent mode to store the reliable messaging sequence information.

- **Non-Persistent** specifies that the reliable messaging sequence information is stored in a non-persistent storage mode. When the **Non-Persistent** mode of storage is used, Integration Server relies on the on-heap memory for reliable messaging data storage. When Integration Server restarts, the reliable messaging information will be removed from memory. This is the default.
- **Database** specifies that the reliable messaging sequence information is stored in a persistent storage mode. When the **Database** mode of storage is used, Integration Server uses a database to store the reliable messaging information. All information related to reliable messaging sequences, including the essential routing and delivery information, is preserved across Integration Server restarts.

Housekeeping Interval (Seconds) Specifies the time interval (in seconds) in which Integration Server sweeps the database to check for timed-out or terminated sequences.

houseKeepingInterval The messages are timed out or terminated depending on the specified **Inactivity Timeout** and **Sequence Removal Timeout** values. Integration Server sweeps the database periodically based on the **Housekeeping Interval** and identifies and marks the messages that are timed out and removes the terminated messages if the `sequenceRemovalTimeout` interval is completed for the sequence.

The default is 20 seconds.

Remote Server Alias

Note:

Deployer does not export remote server aliases that use the alias name “local”.

Asset Type ID isremoteserveralias

Substitution Values

Host Name The host name or IP address of the remote server represented by the alias.

host

Retry Server Host name or IP address of the remote server you want the target Integration Server to connect to if the primary remote server is unavailable.

retryServer

Port The port number that is used by the remote server specified by the alias.

server_port

User Name User name the target server will use to access and invoke services on the remote server.

user

Password Password identified in the user account for **User Name**.

pass

SAML Token Issuer

Asset Type ID issamlissuer

Substitution Values

None.

Scheduled Tasks

Asset Type ID istask

Substitution Values

Run as User User name the target servers will use when running the service.

runAsUser The target servers run the service as if the user you specify is the authenticated user that invoked the service. If the service is governed by an ACL, be sure to specify a user that is allowed to invoke the service.

Cluster Target Node Specifies whether you want the task to run on other target servers in the cluster.

Target

- **Any server** indicates that you want the task to run on only one server in the cluster, and it does not matter which one.
- **All servers** indicates that you want the task to run on all servers in the cluster.
- Enter a specific server name in the cluster if the task needs to run on only a specific server.

Note:

To use this parameter, the source server must be enabled for clustering and the target servers must belong to a cluster.

Suspend During Deployment Specifies whether you want to suspend the existing task during deployment.

suspendDuring
Deploy

- **True** indicates that you want to suspend the task during deployment.
- **False** indicates that you want the task to run as scheduled during deployment.
- **None** indicates that you want to use the global value specified on the **Deployment Map Properties > Configure Builds by Assets** screen and

by the **Suspend tasks during deployment** value in the project map file. The default value is **None**.

Activate after deployment	Specifies whether you want to activate the existing task after deployment.
activateAfterDeploy	<ul style="list-style-type: none"> ■ True indicates that you want the task to activate after deployment. ■ False indicates that you want the task to suspend after deployment. ■ None indicates that you want to use the global value specified on the Deployment Map Properties > Configure Builds by Assets screen and by the Activate tasks after deployment value in the project map file.
	The default value is True if the scheduled task is active (not suspended) on the source server. The default is False if the scheduled task is suspended on the source server.

SFTP Server Alias

Asset Type ID	issftpsserveralias
Substitution Values	
SFTP Client Version	Version 1 or Version 2
Host Name or IP Address	Host name or IP address of the SFTP server.
hostName	
Port Number	Port number of the SFTP server. The port number must be within the range of 0 and 65535 (inclusive).
port	
Preferred Key Exchange Algorithms	The algorithms that Integration Server presents to the SFTP server for key exchange.
Preferred Key Exchange Algorithms	
Proxy Alias	Proxy alias through which the request is to be routed.
proxyAlias	
Host Key Location	Location of the public key file of the SFTP server. Integration Server populates this field with the host key file of the source Integration Server. You can change the value of this field to specify a different host key file for deployment.
hostKey Location	
	<p>Note: The public key file must be present on the same machine in which you have installed Integration Server.</p>

SFTP User Alias

Asset Type ID `issftpuseralias`

Substitution Values

User Name	User name for the SFTP user account.
<code>userName</code>	
Authentication Type	The type of authentication that Integration Server uses to authenticate itself to the SFTP server. Client authentication type can be either <code>password</code> or <code>publicKey</code> .
<code>authenticationType</code>	
Password	Password for the specified user to connect to the SFTP server if you are using password authentication.
<code>password</code>	
PassPhrase	Passphrase for the private key file of the specified user if you are using public key authentication and if the private key you specified requires a passphrase.
<code>passPhrase</code>	
Private Key Location	The location of the private key file of the specified SFTP user if you are using public key authentication. Integration Server populates this field with the private key file of the source Integration Server. You can change the value of this field to specify a different private key file for deployment.
<code>privateKeyFileLocation</code>	
Maximum Retries	The number of times Integration Server attempts to connect to the SFTP server. The maximum allowed value is 6. The minimum allowed value is 1. The default is 6 retries.
<code>maximumRetries</code>	
Connection Timeout	The amount of time (measured in seconds) Integration Server waits for a response from the SFTP server before timing out and terminating the request. The default is 0, which indicates that the session will never time out.
<code>connectionTimeout</code>	
Session Timeout	The number of minutes you want Integration Server to wait before terminating an idle session. The default is 10 minutes.
<code>sessionTimeout</code>	
Compression	Specifies whether or not to compress the data to reduce the amount of data that is transmitted. Integration Server supports compression using the compression algorithm <code>zlib</code> .
<code>compression</code>	

- **zlib** indicates that you want to compress the data that is transmitted between the SFTP server and Integration Server.
- **none** indicates that you do not want to compress the data.

Note:

You can use compression only if the SFTP server that you are connecting to supports compression.

Compression Level	The compression level to use if you specified the compression algorithm <code>zlib</code> for the Compression property. The minimum allowed value is 1 (fast, less compression). The maximum allowed value is 6 (slow, most compression). The default is 6.
<code>compression Level</code>	
SFTP Server Alias	The alias of the SFTP server to which you want the user specified using the User Name property to connect.
<code>sftpServer Alias</code>	

Truststore Alias

Asset Type ID `istruststorealias`

Substitution Values

Location Full path to the Integration Server's client truststore file.

`ksLocation`

Password Password associated with this alias that is used to protect the contents of the truststore.

`ksPassword`

Use Source Truststore Specifies whether the target Integration Server uses the truststore file from the source Integration Server.

`useSource Truststore`

- **True** indicates that the target Integration Server uses the truststore file from the source Integration Server. This is the default.
- **False** indicates that the target Integration Server will use a different truststore file than the source Integration Server.

Note:

This configuration value does not correspond to a field or property in Integration Server.

Universal Messaging Connection Alias

Asset Type ID `isumalias`

Substitution Values

General Settings:

Client Prefix A string that identifies Integration Server to Universal Messaging.

`CLIENT_PREFIX`

Connection Settings:

Realm URL	The URL for the Universal Messaging server in the format: <i>protocol:// UIM_host: UIM_port</i>
um_rname	

Client Authentication Settings:

Username	User name to connect to Universal Messaging.
umUser	

Password	Password required for the specified user name.
umPassword	

Users

Asset Type ID	isuser
----------------------	--------

Substitution Values

None.

Web Service Endpoint Alias

Asset Type ID	iswebserviceendpointalias
----------------------	---------------------------

Substitution Values**Transport Properties:**

Host Name	Host name or IP address of the server on which the web service resides. (Provider and consumer web service endpoint alias only.)
host	

Port Number	Active HTTP or HTTPS type listener port defined on the host server or proxy server. (Provider and consumer web service endpoint alias only.)
port	

User Name	User name used to authenticate the consumer at the HTTP or HTTPS transport level on the host server. (Consumer and message addressing web service endpoint alias only.)
transportUser	

Password	The password used to authenticate the consumer on the host server. (Consumer and message addressing web service endpoint alias only.)
transport Password	

Authentication Type	The type of authentication to authenticate the consumer at the HTTP or HTTPS transport level on the host server. (Consumer and message addressing web service endpoint alias only.)
----------------------------	---

- transportAuth Type ■ **Basic** indicates that basic authentication (user name and password) is used to authenticate the consumer.
- **Digest** indicates that password digest is used to authenticate the consumer.

Message Properties:

User Name The user name to include with the UsernameToken. (Consumer web service endpoint alias only.)

messageUser

Password The password to include with the UsernameToken (must be plain text). (Consumer web service endpoint alias only.)

message Password

Partner's Certificate Path and file name of the file containing the provider's certificate. (Consumer and message addressing web service endpoint alias only)

partner
Certificate
FileName

Use Source Partner's Certificate Specifies whether the target Integration Server uses the partner certificate file from the source.

useSource Partner
Certificate

- **True** indicates that the target server uses the partner certificate file from the source. During deployment, Deployer copies the partner's certificate from the source machine to the location on the destination machine specified in **Partner's Certificate**. This is the default value.

- **False** indicates that the target server will use a different partner certificate file than the source the source partner.

Note:

This configuration value does not correspond to a field or property in Integration Server.

Kerberos Credentials:

JAAS Context The custom JAAS context used for Kerberos authentication.

jaasContext

Client Principal The name of the client principal to use for Kerberos authentication.

client Principal

Service Principal Name The service that the Kerberos client wants to access. This can be obtained from the WSDL document published by the provider of Kerberos service.

service Principal

Service Principal Name Form The format in which you want to specify the principal name of the service that is registered with the principal database, namely, host-based or username.

service
PrincipalForm

Message Addressing Properties:

To toMsgAddr	Endpoint reference to which the SOAP message is sent. (Consumer web service endpoint alias only.)
From fromMsgAddr	Endpoint reference containing the source of the SOAP message. (Consumer and message addressing web service endpoint aliases only.)
ReplyTo replyToMsgAddr	Endpoint reference containing the destination address of the response (reply) message. (Consumer and message addressing web service endpoint aliases only.)
FaultTo faultToMsgAddr	Endpoint reference containing the address to which the SOAP fault messages are routed. (Consumer and message addressing web service endpoint aliases only.)

Reliable Messaging Properties

Note:

The reliable messaging properties apply only to consumer and provider web service endpoint aliases of transport type HTTP and HTTPS.

Enable enable	Whether Integration Server uses the reliable messaging properties specific to the web service endpoint alias or the server-level reliable messaging properties that applies to all web service endpoints in the server. <ul style="list-style-type: none"> ■ True specifies that Integration Server uses the reliable messaging properties specific to the web service endpoint alias. ■ False specifies that Integration Server uses the server-level reliable messaging properties.
-------------------------	---

Note:

Rest of the reliable messaging properties specific to a web service endpoint alias are the same as the server-level reliable messaging properties. For more information about reliable messaging properties, see Reliable Messaging Configuration under [“Integration Server” on page 126](#).

webMethods Enterprise Gateway Configuration

Asset Type ID isenterprisegatewayrules

Substitution Values

Email To emailTo	One or more email addresses to which the Integration Server acting as the Enterprise Gateway Server sends an alert when a request violates an Enterprise Gateway rule. This is the global value associated with the default alert option. The server uses this setting when a rule does not specify a custom alert setting.
User Name userName	Integration Server user ID to run the flow service that executes when a request violates an Enterprise Gateway rule. This is the global value associated with the default alert option. Enterprise Gateway Server uses this setting when a rule does not specify a custom alert setting.
Email To rule_name_emailTo	One or more email addresses to which Enterprise Gateway Server sends an alert when a request violates an Enterprise Gateway rule. This value is part of the custom alert options defined for an individual rule.
User Name rule_name_user Name	Integration Server user ID to run the flow service that executes when a request violates an Enterprise Gateway rule. This value is part of the custom alert options defined for an individual rule.

Web Service Policy

Asset Type ID iswspolicy

Substitution Values

None.

Integration Server Administrative Asset Dependencies

The following table lists the dependent assets and the reference assets that you must include in a deployment set before you can deploy the dependent asset:

Asset	Dependencies
Client certificate	User Note: Default users (for example, Administrator) are not listed as dependencies.
CloudStreams Administration DatabaseConfig	JDBC pool associated with the CloudStreams JDBC Function.
CloudStreams Streaming Providers	OAuth alias identified by the Alias field when Authentication Type is configured as OAuth 1.0 or OAuth 2.0, IS service identified by the ESB Service Name field when Authentication Type is configured as ESB Callback,

Asset	Dependencies
	Truststore identified by the Truststore Alias field and keystore identified by the Keystore alias field.
CloudStreams Streaming Subscribers	Provider identified by the Provider field, IS service identified by the Service Name field, and user name identified by the Run As User field when Destination Type is configured as ESB Service.
CloudStreams Large Data Configuration	<ul style="list-style-type: none"> ■ watt.server.tspace.location ■ watt.server.tspace.max ■ watt.server.tspace.timeToLive
Integration Cloud Accounts	User name identified by the Run As User field.
Integration Cloud Applications	Integration Cloud accounts and Integration Server package assets
JDBC pool alias	JDBC driver alias
JDBC functional alias	JDBC driver alias
LDAP configuration	<p>Group</p> <p>Note: Default groups are not listed as dependencies.</p>
Remote server alias	<p>Keystore alias, ACL</p> <p>Note: Default ACLs (for example, Anonymous) are not listed, and the local remote server alias is not extracted.</p>
SAML token issuer	Truststore alias
URL alias	IS package
Web service endpoint alias	Keystore alias, truststore alias, JMS trigger name (provider JMS), proxy alias (consumer HTTP and HTTPS), keystore alias (consumer HTTPS), JNDI alias (consumer JMS), JMS alias (consumer JMS), web service endpoint alias (provider HTTP, HTTPS, and JMS)

Integration Server Package Assets

This section describes the following:

- [Adding Package Assets to the Source Directory](#)

- [Global Values for Integration Server Package Assets and Composites](#)
- [Individual Values for Integration Server Package Assets](#)

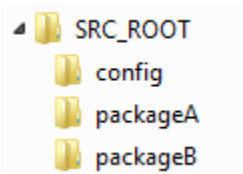
About Integration Server Packages

Integration Server packages can be deployed as either composites or assets with a type ID of `ispackage`.

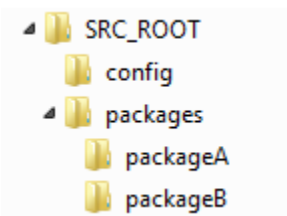
Adding Package Assets to the Source Directory

To include package assets in the composite for deployment, you must manually copy or check in the `Integration Server_directory \instances\instance_name\packages` folder to the source directory. There are two ways to add package assets to the source directory:

- **Build package composites along with other administrative assets.** If you want to generate composites for all packages in the source directory and package-specific administrative assets, Software AG recommends that you structure the source directory as either:



Or:



Note:

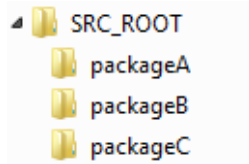
In order to retain ACL information for the package assets, you must add ACL configuration files to the `config` folder.

In the example, you would define `SRC_ROOT` as the value of the `build.source.dir` property in the `build.properties` file. For more information about the `build.properties` file, see [“Setting Build Properties”](#) on page 58.

When run, the build script creates the following:

- A composite named `package_name.zip` for each package included in the source directory, where `package_name` is the name of the package (with a composite type ID of `ispackage`). For this example, the files would be named `packageA.zip` and `packageB.zip`.
- A composite named `isconfiguration.zip` (with a composite type ID of `isconfiguration`) that contains the administrative assets contained in the `config` directory.

- **Build package composites separate from other administrative assets.** If your source directory contains several packages and you want to generate composites from only a select number of those packages, you can use the `build.source.project.dir` property to specify only those packages you want to include. For example, in the following source directory there are three packages: `packageA`, `packageB`, and `packageC`:



You can generate a composite that contains only `packageA` and `packageB` by setting the value of the `build.source.project.dir` property to:

```
SRC_ROOT/packageA;SRC_ROOT/packageB
```

In this example, since the config directory is not located in `SRC_ROOT`, you must specify the location of config directory in the `is.acdl.config.dir` property of the `build.properties` file.

When run, the build script creates a composite named `package_name.zip` for each package defined for `build.source.project.dir`, where `package_name` is the name of the package (with a composite type ID of `ispackage`).

Using the example, `build.source.project.dir` is set to `"SRC_ROOT/packageA;SRC_ROOT/packageB"`. When the build script runs, it will generate composites for `packageA` and `packageB`. Since `packageC` is not defined for `build.source.project.dir`, the build script ignores it. Since the packages are named `"packageA"` and `"packageB"` in the source directory, the build script names the composites `packageA.zip` and `packageB.zip`.

For more information about setting properties in the `build.properties` file, see [“Setting Build Properties” on page 58](#). For more information about building composites for repository-based deployment, see [“Building Composites” on page 57](#).

Global Values for Integration Server Package Assets and Composites

The following table lists the global values you can export for Integration Server packages.

Value	Description
Activate Package on Install	Specifies whether you want the Integration Server to activate the package immediately upon installation.
(<code>activatePkgOnInstall</code>)	<ul style="list-style-type: none"> ■ True indicates that you want the server to activate the package after it is installed. ■ False indicates that you do not want the server to activate the package after it is installed. <p>The default value is True.</p>

Value	Description
Archive Package on Install (archivePkgOnInstall)	<p>Specifies whether you want Integration Server to archive the package automatically after installation.</p> <ul style="list-style-type: none"> ■ True indicates that you want the server to archive the package automatically after it is installed. ■ False indicates that you do not want the server to archive the package after it is installed. <p>The default value is True.</p>
Compile Package on Install (compilePackage)	<p>Specifies whether you want Integration Server to compile the package automatically after installation.</p> <ul style="list-style-type: none"> ■ True indicates that you want the server to compile the package automatically after it is installed. ■ False indicates that you do not want the server to compile the package after it is installed. <p>The default value is True.</p>
Disallow Active Package Install (disallowActivePackageInstall)	<p>Specifies whether you want to prevent deployment if the package being deployed is in an active state on the target Integration Server.</p> <ul style="list-style-type: none"> ■ True indicates that you want to prevent the server from continuing deployment when the package being deployed is in an active state on the target Integration Server. ■ False indicates that you want the server to continue deployment regardless of whether the package is active on the target Integration Server. <p>The default value is False.</p>
Fragment Package on Install (fragPackage)	<p>Specifies whether you want Integration Server to perform a fragmentation step during the compilation of a package. The fragmentation step creates new node.ndf files for any Java services contained in the package. Omitting this step retains the node.ndf files that were copied from the source server, thereby preserving custom settings defined in those files.</p> <ul style="list-style-type: none"> ■ True indicates that you want to allow the server to perform the fragmentation step and overwrite a package's node.ndf files when the package is compiled on the target server. ■ False indicates that you want the server to omit the fragmentation step and retain the node.ndf files that were copied from the source server when the package is compiled on the target server. <p>The default value is True.</p>

Value	Description
Package Execution Check (package ExecutionCheck)	<p>Specifies the length of time (in milliseconds) Deployer should wait if a service contained in the package being deployed is being executed on the target Integration Server. If this time expires and a service is still being executed, Deployer terminates the deployment job. The default value for this parameter is 0, which disables this feature.</p> <p>Note: In some cases, this parameter can override <code>disallowActivePackageInstall</code>. For example, if <code>disallowActivePackageInstall</code> is set to False and <code>packageExecutionCheck</code> is set to a value other than 0, Integration Server can terminate the deployment job even though <code>disallowActivePackageInstall</code> would otherwise allow deployment to succeed.</p>
Suspend Triggers During Deployment (suspend TriggersDuring Deploy)	<p>Specifies whether you want Integration Server to suspend existing triggers before updating them with the deployment.</p> <ul style="list-style-type: none"> ■ True indicates that you want the server to suspend existing triggers before deployment. ■ False indicates that you do not want the server to suspend the triggers. <p>The default value is False.</p>
Synchronize Document Types During Deployment (syncDocTypesTo Broker)	<p>Specifies whether Integration Server should synchronize the publishable document types in the source package with document types on the webMethodes messaging providers (Broker and/or Universal Messaging) that are connected to the target Integration Server.</p> <ul style="list-style-type: none"> ■ True indicates that you want Integration Server to synchronize publishable document types in the target server with the connected messaging providers during deployment. ■ False indicates that you do not want Integration Server to synchronize publishable document types in the target server with the connected messaging providers during deployment. <p>The default value is True.</p> <p>Note: The target Integration Server must be connected to a messaging provider at deployment time for synchronization to occur. If the connected messaging provider is not available, publishable document types are not synchronized for the Integration Server to which the messaging provider is connected. Deployer writes a message to that effect to the deployment report.</p>
Clear ACLs after deployment	<p>Specifies whether to reset ACLs on the assets during deployment.</p>

Value	Description
(clearACL)	<ul style="list-style-type: none"> ■ True indicates that Integration Server will remove any ACLs that are set for the assets. ■ False (the default) indicates that Integration Server should not remove any existing ACLs for the assets.

Individual Values for Integration Server Package Assets

Integration Server supports the following Integration Server package assets for deployment on a project-by-project basis. The values presented in the table are deployed as assets within the ispackage composite. None of the assets support substitutions.

Asset	Asset Type ID	Notes
webMethods messaging triggers	istrigger	None.
C services	iscservice	None.
Document types	isdocumenttype	None.
E-forms	isdocumenttype, isfile	E-forms consist of an IS document and a configuration file. The IS document is handled as a normal IS document asset. Integration Server handles the configuration file as a package file. The IS document does <i>not</i> state a dependency on the config files (they are not associated in Integration Server), so assets that depend on an e-form will need to have dependencies on both the IS document and the configuration file.
Flat file dictionaries	ffdictionary	None.
Flat file schemas	ffschema	None.
Flow services	isflowservice	None.
IS schemas	isschema	None.
Java services	isjavaservice	None.
JMS triggers	isjmstrigger	None.
Package files	isfile	None.
Package folders	isfolder	None.
PRT fragments	isfile	Handled as a package file.

Asset	Asset Type ID	Notes
REST V2 resources	isrestresource	Applies only to the REST resources invoked with the <code>restv2</code> directive. Note: Partial deployment of a package containing this asset is <i>not</i> supported in a runtime-based deployment.
Specifications	isspecification	None.
URL aliases	isurlalias	The URL Alias asset refers to the URL aliases created for the server or a package. The URL Alias asset does not refer to the URL aliases created for services in Designer or Developer. URL aliases for services are deployed along with the service when a package is deployed. The URL aliases for a server are deployed to the WmRoot package.
Web service connectors	iswsconnector	None.
Web service descriptor consumers	iswsdconsumer	None.
Web service descriptor providers	iswsdprovider	None.
XSLT services	isxslt-service	None.

Adapter Runtime and .NET Service Assets

This section describes the adapter runtime (ART) assets and the .NET asset that Integration Server supports for exporting, and their dependencies.

Adding Adapter Runtime and .NET Service Assets to the Source Directory

To include adapter runtime and .NET service assets in the composite for deployment, you must manually copy or check in the *Integration Server_directory* \instances*instance_name*\packages folder to the source directory.

The build script creates a composite named *package_name.zip*, where *package_name* is the name of the package. For example, if the package name is “adapter_service”, the composite name is *adapter_service.zip*. For more information about building composites for repository-based deployment, see [“Building Composites” on page 57](#).

Adapter Runtime Assets

The following table lists the adapter runtime assets and values for all WmART based adapters that you export and deploy using Integration Server:

Asset	Asset Type ID	Substitution Values
Adapter connections	artconnection	<p>A list of connection management properties, and connection properties based on the adapter. The connection properties are a dynamic set and depend on the adapter implementation.</p> <p>Note: Before you deploy an adapter connection, you must set the password for the connection. For more information, see the installation and user's guide for that adapter.</p> <p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment. Valid values:</p> <ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment. ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>
Adapter services	artservice	None.
Adapter listeners	artlistener	<p>Retry Limit (<code>retryLimit</code>). The number of times that the system attempts to start the listener if the initial attempt fails. In particular, this field specifies how many times to retry the <code>listenerStartup</code> method before issuing an adapter connection exception. The default is 5. A value of 0 indicates that the system will make a single attempt.</p> <p>Retry Backoff Timeout (<code>retryBackoffTimeout</code>). The number of seconds the system waits between each attempt to start the listener. The default is 10. The value of this field is ignored if Retry Limit is set to 0.</p> <p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment. Valid values:</p> <ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment.

Asset	Asset Type ID	Substitution Values
		<ul style="list-style-type: none"> ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>
Adapter listener notifications	artlistener notification	<p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment. Valid values:</p> <ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment. ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>
Adapter polling notifications	artpolling notification	<p>Interval (<code>notificationInterval</code>). The time of the polling interval in seconds.</p> <p>Overlap (<code>notificationOverlap</code>). The time the scheduled interval time you set in the Interval field will begin.</p> <ul style="list-style-type: none"> ■ True indicates that the next execution of a scheduled notification does not wait for the current execution to end and they overlap. ■ False indicates that the executions of a scheduled notification do not overlap. <p>Immediate (<code>notificationImmediate</code>). Whether to enable or disable polling immediately.</p> <ul style="list-style-type: none"> ■ True enables polling immediately. ■ False disables polling immediately. <p>State After Deployment (<code>art.deployment.state</code>). The state of the asset after deployment.</p> <ul style="list-style-type: none"> ■ disable indicates that the asset will be in a disabled state after deployment. ■ enable indicates that the asset will be in an enabled state after deployment. Any value other than enable sets the state to disable. <p>The default is disable.</p>

Adapter Runtime Asset Dependencies

The following table lists the dependent assets and the reference assets that you must include in a deployment set before you can deploy the dependent asset:

Asset	Dependency
Adapter Connections	None
Adapter Services	Adapter Connection
Adapter Listeners	Adapter Connection
Adapter Polling Notifications	Adapter Connection, IS Document Type
Adapter Listener Notifications	Adapter Listener, IS Document Type

.NET Asset

The following table lists the .NET asset that you export and deploy using Integration Server and its values:

Note:

The .NET service asset is not a dependent asset. However, ensure that the values in the **Assembly Path** and **Assembly Name** fields that you provide for the target Integration Server during variable substitution are valid.

Asset	Asset Type ID	Substitution Values
.NET Services	dotnetservice	<p>Assembly Path (<code>assemblyPath</code>). The location of the directory that holds the .NET assembly in which the method called by the .NET services resides.</p> <hr/> <p>Assembly Name (<code>assemblyName</code>). The name of the .NET assembly in which the method called by the .NET services resides.</p>

CloudStreams Assets

The following table lists the CloudStreams assets:

Asset	Asset Type ID	Substitution Values
CloudStreams Connector Services	iscloudstreamsconnectorservice	<p>Connector Virtual Service (<code>virtualServiceName</code>).</p> <p>The connector virtual service to be used for policy enforcement.</p> <hr/> <p>Connection Pool (<code>connectionAlias</code>).</p> <p>The connection pool used by the cloud connector service.</p>
CloudStreams Connection	iscloudstreamsconnection	<p>Connection Enabled (<code>connectionEnabled</code>).</p> <p>Status of the connection, post deployment.</p> <ul style="list-style-type: none"> ■ true - Enabled status. ■ false - Disabled status. <hr/> <p>Server URL (<code>cn.providerUrl</code>).</p> <p>The native provider endpoint target for the connection configuration.</p> <hr/> <p>Connection Timeout (<code>cn.connectTimeout</code>).</p> <p>The number of milliseconds a connection attempt will wait before giving up. (0 will wait indefinitely).</p> <hr/> <p>Socket Read Timeout (<code>cn.readTimeout</code>).</p> <p>The number of milliseconds in which the client must read a response message from the server. (0 will wait indefinitely).</p> <hr/> <p>Use Stale Checking (<code>cn.socketStaleCheck</code>).</p> <p>If true, the connection factory performs additional processing to test if the socket is still functional each time the socket is used.</p> <ul style="list-style-type: none"> ■ true - Enable test for functional socket. ■ false - Disable test for functional socket. <hr/> <p>Validate After Inactivity (<code>cn.validateAfterInactivity</code>).</p> <p>This is used to control the period of inactivity in milliseconds after which persistent connections must be revalidated prior to</p>

Asset	Asset Type ID	Substitution Values
		being leased to the consumer. Applicable if "Use Stale Checking" is true. To turn off connection validation, set this field to a non positive value.
		<p>Connection Retry Count (cn.retryCount).</p> <p>Number of times the connection factory should attempt to execute a failed invocation.</p>
		<p>Retry on Response Failure (cn.retryIfRequestSentOk).</p> <p>If true, the retry mechanism will be used for failed responses even if the request was sent successfully.</p> <ul style="list-style-type: none"> ■ true - Enable retries. ■ false - Disable retries.
		<p>Use TCP NoDelay (cn.tcpNoDelay).</p> <p>If true, do not use Nagles algorithm as a socket optimization technique.</p> <ul style="list-style-type: none"> ■ true - Skip socket optimization using Nagles algorithm. ■ false - Use Nagles algorithm for socket optimization.
		<p>Socket Linger (cn.socketLinger).</p> <p>This setting determines how quickly a socket should close.</p>
		<p>Socket Buffer Size (cn.sockBuffSize).</p> <p>The size of the read and write socket buffers in bytes.</p>
		<p>Socket Reuse Address (cn.reuseAddr).</p> <p>If true, the socket will be reused even if it is in TIME_WAIT due to a previous socket closure.</p> <ul style="list-style-type: none"> ■ true - Enable socket reuse. ■ false - Disable socket reuse.

Asset	Asset Type ID	Substitution Values
		<p>Socket Buffer Size (cn.sockBufferSize).</p> <p>The size of the read and write socket buffers in bytes.</p>
		<p>Proxy Server Alias (cn.webproxyAlias).</p> <p>The alias to a web proxy server configuration in Integration Server.</p>
		<p>Trust store Alias (cn.truststoreAlias).</p> <p>Alias for the Integration Server trust store configuration.</p>
		<p>Hostname verifier (cn.hostnameVerifier).</p> <p>Fully qualified class name of the <i>javax.net.ssl.HostnameVerifier</i> implementation. Guards against man-in-the-middle (MITM) attacks.</p> <p>The options are:</p> <ul style="list-style-type: none"> ■ org.apache.http.conn.ssl.DefaultHostnameVerifier ■ org.apache.http.conn.ssl.NoopHostnameVerifier <p>The default selection is <i>org.apache.http.conn.ssl.DefaultHostnameVerifier</i>, which will enable hostname verification. Select <i>org.apache.http.conn.ssl.NoopHostnameVerifier</i> to disable hostname verification.</p>
		<p>Keep Alive Interval (cn.keepAliveInterval).</p> <p>The keep alive interval in milliseconds defines the interval for which a connection will be kept alive, if the back end does not respond with a Keep-Alive header. A value > 0 keeps the connection alive for the specified value. The default value of -1 implies that the connection will be kept alive until a request fails due to a connection error.</p>
		<p>Idle Timeout (cn.idleTimeout).</p> <p>An idle timeout value > 0 defines the time period in milliseconds after which idle or inactive connections will be closed. The</p>

Asset	Asset Type ID	Substitution Values
		<p>default value of -1 implies that idle connections will not be closed.</p>
		<p>Enable Compression (<code>cn.enableCompression</code>).</p> <p>Whether to compress the request using the gzip compression technique.</p> <ul style="list-style-type: none"> ■ true - Request will be compressed using the gzip compression technique. ■ false - Request will not be compressed.
		<p>Username (<code>cr.username</code>).</p> <p>The username credentials for the current connection configuration. This is available only if the Credentials connection group is configured at source.</p>
		<p>Password (<code>cr.password</code>).</p> <p>The password credentials for the current connection configuration. This is available only if the Credentials connection group is configured at source.</p>
		<p>Authorization Type (<code>cr.authSchemeType</code>).</p> <p>The string identifying the authentication protocol scheme to use for the connection configuration. This is available only if the Credentials connection group is configured at source.</p> <ul style="list-style-type: none"> ■ basic - HTTP Basic authentication. ■ none - No authentication scheme.
		<p>Preemptive Auth (<code>cr.preemptiveAuthEnabled</code>).</p> <p>If true, basic auth credentials will be included when a request is sent and it will not wait for a 401 response challenge. This is available only if the Credentials connection group is configured at source.</p> <ul style="list-style-type: none"> ■ true - Include Basic authentication credentials with the request.

Asset	Asset Type ID	Substitution Values
		<ul style="list-style-type: none"> ■ false - Exclude Basic authentication credentials with the request.
		<p>Domain Name (<code>cr.securityRealm</code>).</p> <p>The domain/security realm for the current connection configuration. This is available only if the Credentials connection group is configured at source.</p>
		<p>Keystore Alias (<code>cr.securityRealm</code>).</p> <p>Alias for the Integration Server key store configuration. This is available only if the Credentials connection group is configured at source.</p>
		<p>Client key Alias (<code>cr.clientKeyAlias</code>).</p> <p>Alias to reference a key inside a key store file. This is available only if the Credentials connection group is configured at source.</p>
		<p>HTTP Content Character Set (<code>pr.httpContentCharSet</code>).</p> <p>The encoding to use for the request message. This is available only if the Protocol connection group is configured at source.</p>
		<p>HTTP Protocol Version (<code>pr.protocolVersion</code>).</p> <p>The HTTP version, for example, HTTP/0.9, HTTP/1.0, or HTTP/1.1. This is available only if the Protocol connection group is configured at source.</p> <ul style="list-style-type: none"> ■ HTTP/0.9 ■ HTTP/1.0 ■ HTTP/1.1
		<p>User Agent (<code>pr.userAgent</code>).</p> <p>The value the connection configuration will send for the "User-Agent" request header. This is available only if the Protocol connection group is configured at source.</p>

Asset	Asset Type ID	Substitution Values
		<p>Use Expect Continue (<code>pr.useExpectCont</code>).</p> <p>If true, use the Expect/Continue HTTP/1.1 handshake and send the "Expect" request header. This is available only if the Protocol connection group is configured at source.</p> <ul style="list-style-type: none"> ■ true - Use the Expect/Continue handshake and the Expect header. ■ false - Do not use the Expect/Continue handshake and the Expect header.
		<p>Use Chunking (<code>pr.useChunking</code>).</p> <p>If true, use HTTP/1.1 chunking using a chunk size that matches the socket buffer size. This is available only if the Protocol connection group is configured at source.</p> <ul style="list-style-type: none"> ■ true - Enable chunking. ■ false - Disable chunking.
		<p>Follow Server Redirects (<code>pr.followServerRedirects</code>).</p> <p>If true, follow server redirects. This is available only if the Protocol connection group is configured at source.</p> <ul style="list-style-type: none"> ■ true - Enable server redirects. ■ false - Disable server redirects.
		<p>Server Redirect Maximum Tries (<code>pr.serverRedirectMax</code>).</p> <p>Maximum number of times to follow a server redirect. This is available only if the Protocol connection group is configured at source.</p>
		<p>Request Header Names (<code>rh.requestHeaderNames</code>).</p> <p>An array of request header names to include for this connection configuration. This is available only if the Request Headers connection group is configured at source.</p>

Asset	Asset Type ID	Substitution Values
		<p>Request Header Values (rh.requestHeaderValues).</p> <p>An array of request header values to include for this connection configuration. This is available only if the Request Headers connection group is configured at source.</p>
		<p>Consumer ID (oauth.consumerId).</p> <p>OAuth V1.0a consumer key or OAuth V2.0 client id. This is available only if the OAuth V1.0a or the OAuth V2.0 connection group is configured at source.</p>
		<p>Consumer Secret (oauth.consumerSecret).</p> <p>OAuth V1.0a consumer secret or OAuth V2.0 client secret. This is available only if the OAuth V1.0a or the OAuth V2.0 connection group is configured at source.</p>
		<p>Access Token (oauth.accessToken).</p> <p>OAuth V1.0a request token or OAuth V2.0 access token. This is available only if the OAuth V1.0a or the OAuth V2.0 connection group is configured at source.</p>
		<p>Access Token Secret (oauth_v10a.accessTokenSecret).</p> <p>OAuth V1.0a request token secret. This is available only if the OAuth V1.0a connection group is configured at source.</p>
		<p>Instance URL (oauth_v20.instanceURL).</p> <p>OAuth V2.0 instance URL. This is available only if the OAuth V2.0 connection group is configured at source.</p>
		<p>Refresh Access Token (oauth_v20.refreshAccessToken).</p> <p>Option to refresh the OAuth V2.0 access token. This is available only if the OAuth V2.0 connection group is configured at source.</p> <ul style="list-style-type: none"> ■ true - Enable access token refresh.

Asset	Asset Type ID	Substitution Values
		<ul style="list-style-type: none"> ■ false - Disable access token refresh.
		<p>Refresh Token (<code>oauth_v20.refreshToken</code>).</p> <p>OAuth V2.0 refresh token This is available only if the OAuth V2.0 connection group is configured at source.</p>
		<p>Refresh URL (<code>oauth_v20.accessTokenRefreshURL</code>).</p> <p>URL for the OAuth V2.0 token refresh request. This is available only if the OAuth V2.0 connection group is configured at source.</p>
		<p>Refresh URL Request (<code>oauth_v20.accessTokenRefreshURLRequest</code>).</p> <p>Option for sending the parameters within the OAuth V2.0 token refresh HTTP POST request, either as query parameters, or as body parameters, or as determined by a custom ESB service. This is available only if the OAuth V2.0 connection group is configured at source.</p> <ul style="list-style-type: none"> ■ URL Query String - Refresh the parameters as query parameters. ■ Body Query String - Refresh the parameters as body parameters. ■ Custom ESB Service - Refresh the parameters as formatted by custom ESB service.
		<p>Refresh Custom ESB Service (<code>oauth_v20.accessTokenRefreshCustomESBService</code>).</p> <p>Custom ESB service name, when the option selected for sending the OAuth V2.0 token refresh parameters is "Custom ESB Service". This is available only if the OAuth V2.0 connection group is configured at source.</p>
		<p>Authorization Header Prefix (<code>oauth_v20.authorizationHeaderPrefix</code>).</p>

Asset	Asset Type ID	Substitution Values
		<p>OAuth V1.0a or OAuth V2.0 authorization header prefix. This is available only if the OAuth V1.0a or the OAuth V2.0 connection group is configured at source.</p> <ul style="list-style-type: none"> ■ Bearer - "Bearer" is used as the prefix. ■ OAuth - "OAuth" is used as the prefix.
		<p>OAuth Config Alias (<code>oa.alias_key</code>).</p> <p>The alias to a configured OAuth token. This is available only if the OAuth Alias connection group is configured at source.</p>
		<p>Access Key (<code>aws.accessKey</code>).</p> <p>20 character string representing an AWS user id. This is created at the Amazon web site. This is available only if the AWS V2 Signature, the AWS V4 Signature, or the AWS S3 Signature connection group is configured at source.</p>
		<p>Secret Key (<code>aws.secretKey</code>).</p> <p>40 character private key. This is created at the Amazon web site when the Access Key is created. This is available only if the AWS V2 Signature, the AWS V4 Signature, or the AWS S3 Signature connection group is configured at source.</p>
		<p>Region (<code>aws.region</code>).</p> <p>A specific geographic region. Some AWS services use this to route requests to local servers to reduce the network latency. This is available only if the AWS V2 Signature, the AWS V4 Signature, or the AWS S3 Signature connection group is configured at source.</p>
		<p>Signing Algorithm (<code>aws.signingAlgorithm</code>).</p> <p>Mnemonic for cryptographic algorithm used to produce the signature, for example, HmacSHA1 and HmacSHA256). This is available only if the AWS V2 Signature, the AWS V4 Signature, or the AWS S3 Signature connection group is configured at source.</p>

Asset	Asset Type ID	Substitution Values
		<ul style="list-style-type: none"> ■ HmacSHA256 - Supported algorithm for AWS V2 and AWS V4 Signature. ■ HmacSHA1 - Supported algorithm for AWS V2 and AWS S3 Signature.
		<p>Note: Additionally, any fields configured as part of the Custom connection group that are not marked as hidden, will be available for variable substitution.</p>
CloudStreams Streaming Listener Service	iscloudstreamslistenerservice	<p>connectionAlias - Connection used to create the Listener.</p>
		<p>enabled - Listener State, whether it is enabled or not.</p>
		<p>name - Namespace of the service which is added as a Service Invocation action under the Event tab in Service Development.</p>
		<p>runAsUser - The user, using which the service mentioned in above property needs to be executed.</p>
		<p>level - The logging level for the Log Invocation action.</p>
		<p>Connection Group Properties:</p>
		<p>Streaming Endpoint - The native provider endpoint target for the connection configuration. (<code>hp.endpoint</code>)</p>
		<p>Backoff Increment - The number of milliseconds that the backoff time increments every time a connection with the Bayeux server fails. CometD attempts to reconnect after the backoff time elapses. (<code>bu.backoffIncrement</code>)</p>
		<p>Max Backoff - The maximum number of milliseconds of the backoff time after which the backoff time is not incremented further. (<code>bu.maxBackoff</code>)</p>

Asset	Asset Type ID	Substitution Values
		Max Message Size - The maximum number of bytes of an HTTP response, which may contain many Bayeux messages. (<code>bu.maxMessageSize</code>)

CloudStreams Asset Dependencies

The following table lists the CloudStreams Asset Dependencies:

Value	Description
CloudStreams Connector Services	<ul style="list-style-type: none"> ■ CloudStreams Connection ■ Connector Virtual Service <p>Note: Dependency resolution is enabled only for user-defined connector virtual services.</p>
CloudStreams Connection	<ul style="list-style-type: none"> ■ OAuth alias ■ Proxy Server Alias ■ Trust store Alias ■ Keystore Alias
CloudStreams Listener Service	<ul style="list-style-type: none"> ■ CloudStreams Connection ■ Document type or a service which specifies the event structure ■ Service Invocation action configuration service

Mobile Support

Using Deployer, you can deploy the following webMethods Mobile Support assets:

- Mobile sync component configuration
- Name and version of the mobile application associated with a mobile sync component

The webMethods Mobile Support assets are deployed as assets within the isconfiguration composite when the WmMobileSupport package is installed. The following table lists the dependencies for these assets.

Asset	Asset Type ID	Dependencies
Mobile applications used by Mobile Support	mobileAppSetting	None.
Mobile sync components for Mobile Support	MSCConfigSetting	Ensure that the business document type and the download and upload services specified for the mobile sync components are present on the target server.

Note that the assets in the table should be deployed together. For details about adding these assets to the source directory, see [“Adding Administrative Assets to the Source Directory”](#) on page 127.

My webMethods Server

The following table lists the My webMethods Server (including Task Engine) asset types, as well as the dependencies and properties that you can substitute when using Deployer to deploy assets to a My webMethods Server.

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Access privilege	AccessPrivilege	The rights of a user, group, or role to view applications and features in the Navigation panel and to access pages associated with them. See Functional privileges. Dependencies: The user, group, or role specified by the access privilege.
Business calendar	Calendar	A global calendar used in My webMethods Server (for example, a US holidays calendar).
CAF application	CAFApp	A Composite Application Framework application. Substitution variables: Any env-properties or configuration properties from the application’s web.xml file (for example, properties dynamically added when you create a web connector for your portlet).
CAF runtime configuration	RuntimeConfig	The runtime configuration for a Composite Application Framework application, as modified in the CAF Runtime Administration page. Substitution variables: Any env-properties or configuration properties from the application’s web.xml file (for example, properties dynamically

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Certificate	Certificate	<p>added when you create a web connector for your portlet).</p> <p>A digital certificate associated with an My webMethods Server user.</p>
Component	Component	<p>A legacy portlet or Dynamic Business Object (DBO).</p> <p>Dependencies: Any portlets or DBOs referenced by the xmlImport.xml file of the Component.</p>
Data source	Datasource	<p>An external data source configuration (for example, a connection to an external Oracle database).</p> <p>Substitution variables: The user name, password, and URL for the data source.</p>
Directory service	Directory	<p>Configuration of LDAP or Database directory service from My webMethods Server.</p> <p>Dependencies:</p> <ul style="list-style-type: none"> ■ LDAP directory service - None ■ Database directory service - Data Source <p>Substitution variables: The user name, password, and LDAP URL for the LDAP directory service only. There are no substitution variables for the database directory service.</p>
Functional privilege	FuncPrivilege	<p>The rights of a user, group, or role to make changes within an application or feature, such as to create and modify a workspace. You can export all of the functional privileges associated with a specific user, group, or role. See Access Privileges.</p> <p>Dependencies: Users, groups, or roles to which functional privileges are granted.</p>
Group	Group	<p>A collection of users and other groups. Groups are defined and stored in an internal system or external directory service.</p> <p>Dependencies: The external directory service where the groups are defined. Only groups from internal "System" directory service can be deployed. Groups from LDAP and Database directory come implicitly from the directory itself.</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Locale rule	Rule.locale	<p>A rule configuration that defines the locale (language and country code) to use when locale information is not specified in the user profile.</p> <p>Dependencies: Any users, groups, or roles referenced by the rule logic.</p>
Login page rule	Rule.login	<p>A login page rule configuration that defines the login page to be used.</p> <p>Dependencies: Any users, groups, or roles referenced by the rule logic of the login page that is the target of the rule.</p>
Page/Folder	Folder	<p>The layout and content of a folder or page.</p> <p>Dependencies: The users, groups, and roles referenced in the folder access control. In addition, folders and pages have a dependency on the portlets and Composite Application Framework applications included in the page.</p>
Portlet	Portlet	<p>A portlet is a mini-application or a piece of functionality that can be part of a Composite Application Framework application or can reside independently on a page on the server.</p>
Rendering rule	Rule.render	<p>The user interface formatting capabilities assigned to specific server objects by defining rendering rules. Renderer rules determine the specific renderer to use.</p>
Role	Role	<p>A collection of users assigned to a specific role defined for any directory service.</p> <p>Dependencies: Users, groups, or roles contained within the role.</p>
Saved search	Search	<p>A saved query associated with a particular search page within My webMethods Server (for example, Task List Management or My Inbox). A Saved Search asset includes all saved searches for a particular search page (not individual searches).</p>
Security realm	Realm	<p>A way to apply an access control list to a list of objects. You can organize Security Realms into containers.</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		Dependencies: Users, groups, or roles, and the pages and folders controlled by the security realm.
Shell	Shell	An installable component used to display the header, footer, and title bars for pages. Dependencies: Any custom portlets referenced by the shell section pages.
Shell rule	Rule.shell	A rule configuration that determines what shell should be displayed for each page. Dependencies: Any users, groups, or roles referenced by the rule logic of the shell that is the target of the shell rule.
Skin	Skin	An installable component that defines the look and feel of the rendered page. A skin modifies the images, fonts, colors, and other subtle style aspects of HTML content, without functionally modifying the HTML content.
Skin rule	Rule.skin	A rule configuration that determines what skin should be displayed. Dependencies: Any users, groups, or roles referenced by the rule logic of the skin that is the target of the skin rule.
Start page rule	Rule.startpage	A rule that determines which page is displayed after a user logs into the server. Dependencies: Any users, groups, or roles referenced by the rule logic of the page or folder that is the target of the rule.
Task	Task	Represents human activity for a BPM ProcessModel. Tasks are composed of Task Rules and portlets that implement UIs of the task. Dependencies: <ul style="list-style-type: none"> ■ Task Rule ■ Portlets ■ Business Rules (for assignments)
Task rule	Rule.task	A rule within a task type to specify task assignments and behaviors.

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
User	User	Dependencies: Any users, groups, or roles referenced by the rule logic. An individual listed in the internal “system” directory service and all profile and preference attributes.

Optimize

webMethods Optimize enables you to export assets such as dimensions, event maps, dimension hierarchies, data filters, process configurations, intelligent links, or rules to Deployer. Using Deployer, you can deploy your Optimize assets to another Optimize Analytic Engine and share the assets that you created in Optimize with other applications.

The following table lists the asset types that you can export from Optimize and their dependencies.

Asset	Asset Type ID	Dependencies
Custom trees	OptimizeCustomTree	KPIs, Process models
Data filters	OptimizeDataFilter	Dimensions
Dimensions	OptimizeDimension	Intelligent links
Dimension hierarchies	OptimizeHierarchy	Dimensions
Event maps	OptimizeEventMap	Dimensions, Intelligent links
Intelligent links	OptimizeILink	None
KPIs	OptimizeKpi	Dimension hierarchies, event maps You cannot deploy individual KPIs. Instead, Deployer deploys all KPIs for a selected event map.
Process configurations	OptimizeProcessConfig	Process models
Rules	OptimizeRule	Event maps, Data filters, KPIs

Optimize-Specific Build Properties

When building composites from Optimize assets, set the following Optimize-specific property in the `master_build/build.properties` file.

optimize.acdl.validation

Specifies whether to switch the XSD validation of the generated composite and descriptor files. Set to:

- on to use XSD validation.
- off to turn off XSD validation. This is the default.

Trading Networks

The following table lists the Trading Networks asset types you can deploy and their dependencies.

Note:

Asset	Asset Type ID	Dependencies, Substitution Variables
Archive schedules	archiveschedule	Dependencies: <ul style="list-style-type: none"> ■ Partner profiles ■ BizDocType Substitution Variables: Archive Schedule Status - The values are: <ul style="list-style-type: none"> ■ enabled ■ suspended
Binary types	binarytype	None
Contact types	contacttype	None
Data permissions (DLSs)	dls	Dependencies: <ul style="list-style-type: none"> ■ Partner profiles ■ Document types ■ Processing rules ■ My webMethods roles
Document attributes	documentattribute	Dependencies: None Substitution Variables: Document Attribute Status - The values are: <ul style="list-style-type: none"> ■ enabled ■ disabled

Asset	Asset Type ID	Dependencies, Substitution Variables
Document types	documenttype	Dependencies: <ul style="list-style-type: none">■ Document attributes■ ESB services Substitution Variables: <p>Document Type Status - The values are:</p> <ul style="list-style-type: none">■ enabled■ disabled
External ID types	externalidtype	None
Field definitions	fielddefinition	Dependencies: Field groups
Field groups	fieldgroup	None
General functional permissions	functionalpermission	Dependencies: My webMethods roles
Partner profiles	partner	Dependencies: <ul style="list-style-type: none">■ Contact types■ External ID types■ Profile groups■ Field definitions■ ESB services Substitution Variables: <ul style="list-style-type: none">■ Delivery Method Name: Host■ Delivery Method Name: Port■ Delivery Method Name: Username■ Delivery Method Name: Password■ Partner Profile Status<ul style="list-style-type: none">■ active■ inactive
Processing rules	processingrule	Dependencies: <ul style="list-style-type: none">■ ESB services

Asset	Asset Type ID	Dependencies, Substitution Variables
		<ul style="list-style-type: none"> ■ Partner profiles ■ Document attributes ■ Document types ■ Profile groups <p>Substitution Variables:</p> <p>Processing Rule Status - The values are:</p> <ul style="list-style-type: none"> ■ enabled ■ disabled
Profile groups	profilegroup	None
Queues	queue	<p>Dependencies: ESB services</p> <p>Substitution Variables:</p> <p>Queue Status - The values are:</p> <ul style="list-style-type: none"> ■ enabled ■ disabled ■ draining ■ suspended
Trading partner agreements (TPAs)	tpa	<p>Dependencies:</p> <ul style="list-style-type: none"> ■ Partner profiles ■ IS document types ■ ESB services <p>Substitution Variables:</p> <p>TPA Status - The values are:</p> <ul style="list-style-type: none"> ■ agreed ■ proposed ■ disabled

Usage Notes

- The values you specify for the substitution variables are case sensitive.

- To track and monitor the deployment of Trading Networks assets, make sure that audit logging is enabled in My webMethods for the Trading Networks logs.

Task Engine

You can deploy Task Engine assets only to Integration Server runtimes. The following table lists the asset that you can deploy to Task Engine servers using Deployer.

Asset	Asset Type ID	Dependencies
Task Application	TaskEngineApp	None.

Universal Messaging

This following table lists the Universal Messaging assets you can deploy and the dependencies and substitution variables (if applicable) of each asset.

You can modify some of the listed assets during deployment if an asset with the same ID already exists on the target server, but others you cannot modify. The last column in the table will state which of the assets cannot be modified, and in that case you must use a deletion set to delete the asset from the target server before deploying. For more information about deleting assets from target servers, see [“Adding Assets to a Repository-Based Deletion Set” on page 96](#).

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
Realm ACLs	RealmAcl	Realm ACL assets can consist of either ACLs in <i>user@host</i> format or groups that contain sets of ACLs. Deployer always replaces target assets. Dependencies: Security groups
Security groups	RealmSecurityGroup	If a security group asset of the same name exists on the target server, you must delete the security group from the target server before deploying.
Realm schedules	RealmSchedule	If a realm schedule asset of the same name exists on the target server, you must delete the realm schedule from the target server before deploying.
Realm configurations	RealmConfig	Deployer always replaces target assets.
Channels	Channel	If the asset exists on the target server, only the ACL can be modified during deployment. If you want to modify

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		<p>additional properties, you must delete the channel asset on the target server before deploying.</p>
Channel joins	ChannelJoin	<p>Deployer always replaces target assets.</p> <p>Dependencies: Source channel and target queue/target channel</p>
Queues	UMQueue	<p>If the asset exists on the target server, only the ACL can be modified during deployment. If you want to modify additional properties, you must delete the queue asset on the target server before deploying.</p>
Interfaces	Interface	<p>The interface asset can contain four types of interfaces: nsp, nsps, nhp, and nhps. For all of the interfaces, Deployer modifies the VIA list on the target server. The nhp and nhps interfaces can contain plugins, which can also be modified.</p> <p>If an interface asset of the same name exists on the target server, you must delete the interface from the target server before deploying.</p> <p>Substitution values: Available for nsps and nhps interfaces only:</p> <ul style="list-style-type: none"> ■ Keystore ■ Keystore Password ■ Truststore ■ Truststore Password ■ Private Key Password
Data groups	DataGroup	<p>Deployer modifies publishers and nested groups on the target server.</p> <p>If a data group asset of the same name exists on the target server, you must delete the data group from the target server before deploying.</p>

Asset	Asset Type ID	Dependencies, Substitution Values, and Other Considerations
		Dependencies: Nested groups
Clusters	Cluster	DEPRECATED. Cluster configuration is deprecated as a deployable asset type. Use the Universal Messaging cluster management function in Software AG Command Central.
Connection Factories	ConnectionFactory	If the asset exists on the target server, you cannot replace or modify the asset. If you want to replace the asset in the target server, you must first delete the existing asset in the target server.
Topic Connection Factories	TopicConnectionFactory	
Queue Connection Factories	QueueConnectionFactory	
XA Connection Factories	XAConnectionFactory	
JNDI Topics	JNDITopic	
JNDI Queues	JNDIQueue	

Other Product Assets

The following table describes additional user-created assets that you can include in deployment projects and their dependencies.

Asset	Asset Type ID	Dependencies
BAM process models	bamprocess	None

6 Managing Deployment Projects

■ About Configuring Project Settings	202
■ Settings for All Projects	202
■ Settings for Runtime-Based Deployment Projects	202
■ Settings for Repository Based Deployment Projects	207
■ Creating a Project	209
■ Exporting and Importing Project Properties	212
■ Managing User Permissions for Project Tasks	213
■ Adding and Viewing Instructions or Notes About a Project	215
■ Update the Settings for a Project	215
■ Deleting a Project	216

About Configuring Project Settings

You configure the default settings for Deployer projects from **Deployer > Settings**. Some of the settings apply for all types of projects, other settings are restricted to a specific project type, for example only for run-time based deployment or only for repository-based deployment.

When creating a project or even after a project is created, you can override many of the default settings for Deployer projects.

Settings for All Projects

Project Locking Default

Indicates whether locking is enabled or disabled for all projects. If locking is enabled, you must lock the project before you can modify the project or perform an action, for example deploy the project. When a project is locked, other users will be able to view the project and run display commands. If necessary, users with administrative privileges can unlock a locked project.

General Deployment Defaults

Setting	Description
Enable Concurrent Deployment	To set Deployer to deploy assets concurrently, click Yes (the default). If you want to deploy assets sequentially, click No . For more information about concurrent and sequential deployment, see “Concurrent and Sequential Deployment” on page 13 .
Maximum Plugin Objects Displayed	Sets the maximum number of assets Deployer displays in the tree for deployment and deletion sets. The default is 2500.

Settings for Runtime-Based Deployment Projects

Dependency Checking Default

Determines the default behavior of Deployer when checking dependencies.

Setting	Description
Always	Tells Deployer to automatically check dependencies regularly as you modify the project and progress through the different phases of deployment.
Reduced	Tells Deployer to automatically check dependencies when you create a project build and when you deploy. You can trigger additional dependency checking at different steps yourself.

Setting	Description
Manual	You will trigger dependency checking at different steps yourself.

General Deployment Defaults

Setting	Description
Large File Support	<p>For Integration Server packages and webMethods files, indicates whether Deployer will stream from the source server to Deployer and from Deployer to the target server by default.</p> <p>Streaming is available only over HTTP. If your source or target environment uses HTTPS, you cannot choose to stream.</p> <ul style="list-style-type: none"> ■ If you choose not to stream, the build size of a project containing Integration Server packages and webMethods files cannot exceed the amount of RAM configured for the source or target Integration Server or Deployer. If it does, you will have to distribute the Integration Server packages and webMethods files across multiple projects (and thus multiple project builds) instead. ■ If you choose to stream, the build size of a project containing Integration Server packages and webMethods files can be up to 4GB. For streaming to work, you must set certain extended settings on every source and target Integration Server, and on the Integration Server that hosts Deployer. <p>In Integration Server Administrator, go to the Settings > Extended > Edit Extended Settings page. Type the following server configuration parameters and values in the box.</p> <ul style="list-style-type: none"> ■ <code>watt.server.tspace.timeToLive=<i>time</i></code> ■ <code>watt.server.tspace.location=<i>full_path_on_local_machine</i></code> ■ <code>watt.server.tspace.max=<i>maximum</i></code> <p>If your project contains BPM process models or Integration Server packages, set the following server configuration parameters as well:</p> <ul style="list-style-type: none"> ■ <code>watt.server.SOAP.MTOMStreaming.enable=true</code> ■ <code>watt.server.SOAP.MTOMStreaming.cachedFiles.location=<i>directory_path</i></code> ■ <code>watt.server.SOAP.MTOMStreaming.threshold=<i>number_of_bytes</i></code> <p>After saving the changes, you must restart Integration Server.</p>
Optimize UI Response	<p>Bypasses the ping requests from the GUI that Deployer usually performs during the map and define dependency checks. The default setting is No. If set to Yes, the responsiveness of the Deployer GUI for large projects gets better.</p>

Setting	Description
Checkpoint Creation	<p>To have Deployer automatically create a checkpoint for target servers before deploying, click Automatic. If you want to generate checkpoints when you choose, click Manual.</p> <p>Note: If deployment sets in a project include deletion sets, Deployer ignores these settings. Instead, Deployer automatically creates checkpoints for target servers.</p>
Rollback on Error	<p>To have Deployer automatically create checkpoints before deploying and roll back deployments if they fail, click Automatic. If you want to roll back deployments when you choose, click Manual.</p> <p>Note: If deployment sets in a project include deletion sets, Deployer ignores these settings. Instead, Deployer automatically rolls back the target servers.</p>
DeployerService Timeout	<p>The amount of time (in milliseconds) to wait before Deployer services time out waiting for a response.</p> <p>This setting overrides the <code>watt.net.timeout</code> and <code>watt.server.SOAP.request.timeout</code> server configuration properties for Deployer.</p> <p>Note: When using the Integration Server or BPM server ping operations, Deployer calls a service that returns cluster information. If the target Integration Server or BPM server is down, this service can take a long time to time out. To reduce the amount of time to wait, set the <code>clusterServiceInvokeTimeout</code> property in the <code>WmDeployer/config/deployer.cnf</code> file. The default value is 2000 milliseconds. Change this value according to your system's requirements and reload the Deployer package for the changes to take effect.</p>
Batch Size for Runtime Deployment	<p>Sets the number of assets that Deployer deploys at one time for runtime-based deployment. Enter the maximum number of assets to build and deploy as a batch at one time. This limits the number of assets in each deployment to only the number indicated. The default value is 10.</p> <p>By batching deployments, you break the transportation of large assets into smaller batches, which decreases the memory utilization.</p> <p>If you set this property to 0, then Deployer does not limit the number of assets during build, checkpoint, rollback, and deploy operations.</p>

IS & TN Deployment Defaults

You can set the following default settings for all Integration Server and Trading Networks projects.

These default settings apply to all assets except Integration Server packages. You specify package properties for Integration Server packages on a package-by-package basis. For instructions, see [“Setting Package Properties” on page 39](#).

Suspend During Deployment

Indicates whether Deployer should suspend activity for the Integration Server assets while deployment is going on. Typically, if the targets are production Integration Servers, you would suspend all of these types of assets. After deployment, Deployer enables the disabled ports and resumes the suspended triggers, adapter listeners, polling notifications, and scheduled tasks.

Asset	Description	Value
Triggers	Allow all running trigger operations to complete, then suspend all trigger execution and document retrieval on the target Integration Servers.	All
	Note: If you choose All , Deployer suspends execution and document retrieval for ALL triggers on the target Integration Servers, not just for the triggers that you include in the project.	
	Do not suspend triggers.	None
	Suspend individual triggers. You choose the triggers to suspend when you set package properties (see “Setting Package Properties” on page 39).	Selected
Ports	Whether to disable ports on the target Integration Servers that match ports you are trying to deploy.	
Scheduled Tasks	Whether to prevent scheduled tasks on the target Integration Servers that match scheduled tasks you are trying to deploy from running.	
	Note: Tasks that are already running at deployment time are not affected by deployment.	
Adapters	Do not suspend adapter listeners or polling notifications.	None
	Suspend individual adapter listeners and polling notifications. You choose the notifications to suspend when you set package properties (see “Setting Package Properties” on page 39).	Selected

Overwrite Existing

Indicates how Deployer should proceed when it finds that assets you are trying to deploy already exist on target Integration Servers.

Asset	Indicate whether Deployer should	Value
TN Rules	Replace the entire rule list.	Replace All
	Overwrite existing rules and deploy new rules into the rule set.	Merge
ACL Maps	Deploy the mapping of ACLs to services for any services you choose to deploy. Deploy ACL maps if you want to assign the same ACLs to the deployed services on the target Integration Server that you assigned to the source services on the source Integration Servers.	
Other Non-Package Assets	Overwrite existing assets. This option applies to all assets except the following: <ul style="list-style-type: none"> ■ Trading Networks processing rules (see the previous step). ■ Integration Server ACL maps (see ACL Maps). ■ Integration Server packages. You specify the overwrite option for Integration Server packages on a package-by-package basis, as described in “Setting Package Properties” on page 39. 	
Scheduled Tasks By	This property can be set to either Service Name or ID . If this property is set to Service Name , and if a scheduled task with the same service name already exists, Deployer overwrites it with the one being deployed. If this property is set to ID , then based on the task ID which is present on the target, Deployer either overwrites the task if the source and target task IDs are same, or creates a new one.	

Note:

Before you deploy a project for runtime-based deployment, you can find out which assets Deployer will overwrite by generating the simulation report.

Activate After Deployment

Indicates whether Deployer should activate the following assets on the target Integration Servers. Configure this setting *only* if **Suspend During Deployment** is set to **Yes**.

Asset	Indicate whether Deployer should
Ports	Activate newly deployed ports.
	Note:

Asset	Indicate whether Deployer should
	If you choose to activate ports, and one of the ports you deploy uses the same port number as an existing port on a target Integration Server, Deployer will display a message to that effect and will not activate the port.
Scheduled Tasks	Activate newly deployed scheduled tasks.
Adapters	Activate newly deployed adapter connections, notifications, and listeners. Note: If you choose to not activate, the deployed adapter connections, notifications, and listeners will be in the same state they are in on the source server.

Packages

Indicates whether Deployer should compile the package and/or reload the package on the target Integration Server.

For this option...	Indicate whether Deployer should...
Compile Java Services	Compile the package during deployment. When Integration Server packages contain Java sources and are deployed using runtime-based deployment, the Java sources will get compiled on the target Integration Server if the Compile Java Services property is set to Yes .
Reload Packages	If the Reload Packages option is set to Yes , deployed packages get reloaded after the package is activated. If the property is set to No , packages will not be reloaded after deployment. The default value is No .

Settings for Repository Based Deployment Projects

General Deployment Defaults

Setting	Description
Batch Size for Repository Deployment	Sets the number of assets that Deployer deploys at one time for repository-based deployment. Enter the maximum number of assets to build and deploy as a batch at one time. This limits the number of assets in each deployment to only the number indicated. The default value is 1. Note: If you are deploying BPM or Optimize assets, set this parameter to 0.

Setting	Description
DeployerService Timeout	<p>The amount of time (in milliseconds) to wait before Deployer services time out waiting for a response.</p> <p>This setting overrides the <code>watt.net.timeout</code> and <code>watt.server.SOAP.request.timeout</code> server configuration properties for Deployer.</p> <p>Note: When using the Integration Server or BPM server ping operations, Deployer calls a service that returns cluster information. If the target Integration Server or BPM server is down, this service can take a long time to time out. To reduce the amount of time to wait, set the <code>clusterServiceInvokeTimeout</code> property in the <code>WmDeployer/config/deployer.cnf</code> file. The default value is 2000 milliseconds. Change this value according to your system's requirements and reload the Deployer package for the changes to take effect.</p>

Preparing Integration Server to Stream Large Repository-Based Projects

Note:

For information about streaming large runtime-based projects, see the description of the **Large File Support** option in [“Settings for Runtime-Based Deployment Projects” on page 202](#).

If you choose to stream repository-based projects from Deployer to the target server, you must set certain server configuration settings on every target Integration Server hosting the runtime and Deployer. The build size of a project containing Integration Server packages and webMethods files can be up to 4GB.

You can stream assets from Deployer to target servers for the following runtime types:

- Integration Server
- Trading Networks
- BPM process models
- EDA
- Event Server

Note:

Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

> To set server configuration parameters

1. For every target Integration Server hosting the runtime type and Deployer, open the Integration Server Administrator and go to the **Settings > Extended > Edit Extended Settings** page.

2. Type the following server configuration parameters and values in the box. For complete information about these server configuration parameters, see the *webMethods Integration Server Administrator's Guide*.
 - `watt.server.SOAP.MTOMStreaming.enable=true`
 - `watt.server.SOAP.MTOMStreaming.cachedFiles.location=directory_path`
 - `watt.server.SOAP.MTOMStreaming.threshold=number_of_bytes`
3. Click **Save Changes** and restart Integration Server.

Creating a Project

You can create a project by creating a new, blank project or by copying an existing project and modifying it.

» To create a project

1. Go to the **Deployer > Projects** page.
2. Create a project using one of these methods:
 - To create a new project:
 1. Click **Create Project**.
 2. In the **Name** box, type the name to use for the new project. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:
`$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "`
 3. In the **Description** box, type a description for the project. The description length has no limit and can include any characters.
 4. In the **Project Type** area, select one of the following:

To create a...	Click...
Runtime-based deployment project	Runtime
Repository-based deployment project	Repository

5. Click **Create**.
- To create a project from an existing project:
 1. Click **Copy Project**.
 2. In the **Project to Copy** box, click the project to copy.

3. In the **New Project Name** box, type the name to use for the new project. The name can be up to 32 characters long and can include any characters that are valid for a file name in your operating system.
 4. Click **Copy Project**.
3. Review the default properties for projects in the right-hand pane and override for the project you are creating if necessary.

In the right-hand pane, Deployer displays those properties for which you can override the settings for an individual project. By default, these properties inherit the values set for the default as described in [“Settings for All Projects” on page 202](#). For example, if the global property for the **Enable Concurrent Deployment** property is set to **Yes** (for concurrent deployment), you can set an individual project to use sequential deployment by setting this property to **No**.

4. Depending on the specific type of project you are creating, you can set the following additional properties in the right-hand pane:
 - If you are creating a repository-based project, you can set the following additional properties:

For this property...	Indicate whether Deployer should...
Enable Project Locking	Whether locking is enabled or disabled for the project. Click Yes to enable locking. Click No to disable locking.
Enable Concurrent Deployment	Deploy assets concurrently. Click Yes to deploy assets concurrently. If you want to deploy assets sequentially, click No . For more information about concurrent and sequential deployment, see “Concurrent and Sequential Deployment” on page 13 .
Ignore Missing Dependencies	Ignore missing dependencies. If you click Yes , Deployer deploys the composite even when the dependent composite is not available on the repository.
Enable Transactional Deployment	<p>Automatically create a checkpoint prior to delivering and activating deployment and deletion sets. If set to Yes (the default), transactional deployment is enabled.</p> <p>When transactional deployment is enabled and activation fails, Deployer triggers a roll back automatically and restores the target servers to the state of the prior activation.</p>

- If the project is for IS & TN, see [“Settings for Runtime-Based Deployment Projects” on page 202](#).

- If the project is for Optimize, you can set the following properties under the **OptimizeOptions** area:

Note:

The following properties are available only for Deployer 8.2 SP1 and earlier.

For this property...	Indicate whether Deployer should...
Include Dimension Values	Indicates whether Deployer should include the values for dimensions you add to deployment sets (for example, Customer Names or Product Types).
Display Data Definition Statements	Indicates whether Deployer should display the values for data definition statements in the binary stored in the project build.

- If the project is for process models, you can set the following properties for the project under the **ProcessModel Deployment Options** area. For more information about process models, see *webMethods Monitor User's Guide*.

For this property...	Indicate whether Deployer should...
Enable process for execution	<p>Enable webMethods-executed business process versions for execution after deployment. When a process version is enabled, the Process Engine uses the enabled version when starting new process instances. When a process is disabled, the Process Engine does not use the process version for new process instances.</p> <p>Only one version of a process can be enabled at a time. If there are no enabled process versions, the Process Engine will not start any process instances of the process.</p>
Enable process for analysis	Enable webMethods-executed processes for analysis after deployment. When a process is enabled, the Process Engine forwards all process instance activity to the Optimize Analytic Engines. When a process is disabled, no activity is forwarded.

- If the project is for My webMethods Server, you can set these properties for the project under the **MWS Deployment Options** area:

For this property...	Indicate whether Deployer should...
Export Subscriptions	Deploy subscriptions for My webMethods Server assets you are deploying.

For this property...	Indicate whether Deployer should...
Export Access Control Lists	Deploy ACLs for My webMethods Server assets you are deploying.
Export Principal Attributes	Include attributes contained in attributes providers when exporting users, groups, and roles.
Export Content As Reference	Export a reference to the page content without deploying the content.
Alias Prefix	Apply the specified prefix to every automatically generated My webMethods Server alias.
Export Version History	Include all versions of an asset in Portal version control. This applies to the content within a page or folder.
Auto Generate Aliases	Automatically generate an alias on the target My webMethods Server for every My webMethods Server asset that is deployed. If an asset already has one or more aliases, then the aliases are retained when the auto-generated alias is added.
Export Content (Documents)	Deploy content referenced by portal pages and folders you are deploying (for example, a PDF document that has been published on a portal page you are deploying).
Page Depth	<p>If the value of this property is 1, all first level child pages that are under a selected parent page are deployed, even if the child pages are not selected in the deployment set. Default value is 1.</p> <p>If the value of this property is 0, only those child pages that are selected will be deployed. Child pages that are not selected will not be deployed.</p>

5. Click **Save**.

Exporting and Importing Project Properties

You can export and import project properties for deployment projects.

Note:

Deployer does not export and import deletion sets as a part of this procedure. For more information about exporting and importing deletion sets, see [“Exporting and Importing Deletion Set Definitions” on page 50](#).

When you export project properties, Deployer creates a file that contains the project property settings. The file is named *project.properties* and is stored in the *Integration Server_directory* \packages\WmDeployer\replicate\outbound directory. You can then import the project property settings into another Deployer project.

> To export and import the project properties

1. Export project properties as follows:
 - a. In Deployer, go to the **Deployer > Projects** page.
 - b. In the **Name** column, click the project from which to export.
 - c. In the right-hand pane, click **Export Project properties**. Deployer creates a file that contains the project property settings. The file is named *project.properties* and is stored in the *Integration Server_directory \packages\WmDeployer\replicate\ outbound* directory. Deployer also gives you the option to save the file to your local file system.
2. Import project properties into another project as follows:
 - a. Copy the *project.properties* file to the *Integration Server_directory \packages\WmDeployer\replicate\inbound* directory on the machine that hosts the project into which to import.
 - b. In Deployer, go to the **Deployer > Projects** page.
 - c. In the **Name** column, click the project into which to import.
 - d. In the right-hand pane, click **Import Project properties**, then select the *project.properties* file you just copied to the inbound directory.

Managing User Permissions for Project Tasks

You can authorize users to perform tasks by project. To do this you use tasks and authorizing groups or My webMethods Server central user management. This means that when Deployer users display the **Projects** page, they will see only those Deployer projects for which they are authorized.

You can authorize groups to perform the following tasks:

- **View.** View projects only.

Note:

Users with Developer and Internal ACLs and any combination of Define, Build, Map, or Deploy authorization automatically have the View authorization.

- **Define.** Groups can define, export, and import deployment and deletion sets only.
- **Build.** Groups can build projects only.
- **Deploy.** Groups can deploy deployment or deletion sets only (that is, to actually deploy assets to or delete assets from target servers or target groups).

- **Map.** Groups can map projects to repositories, source servers, target servers, and target groups.

You grant privileges to perform tasks through Access Control Lists (ACLs). When an administrator creates ACLs, he or she identifies groups that are allowed to perform particular tasks. The following table shows the ACLs and authorizations required to perform each task:

	All Project Tasks	View Project Tasks	Define Project Tasks	Build Project Tasks	Map Project Tasks	Deploy Project Tasks
ACLs						
Developer	X	X	X	X	X	X
Internal	X	X	X	X	X	X
DeployerAdmin	X*					
Administrator	X*					
Authorizations						
View		X				
Define			X			
Build				X		
Map					X	
Deploy						X

Note:

The asterisk (*) indicates that only users with either one of the following ACL combinations can perform all project tasks in Deployer:

- DeployerAdmin, Developer, and Internal ACLs
- Administrator ACL

Users assigned to one of these ACL combinations do not require authorization to specific tasks.



You can authorize groups to perform more than one task. For example, if you want to allow Group A to map and deploy projects, you would select the Map and Deploy authorizations.

Note:


The groups you create for use with Deployer can contain unique names to help define which tasks each group can perform. For example, you could create groups named `viewDeployerProjects`, `buildDeployerProjects`, `mapDeployerProjects`, `deployDeployerProjects`, and `defineDeployerProjects`. This means that when Deployer users display the **Projects** page, they will see only those Deployer projects to which they are authorized.

For more information about groups and ACLs, see *webMethods Integration Server Administrator's Guide*. For information on My webMethods Server central user management groups, see *Administering My webMethods Server*.

➤ To authorize groups to perform tasks

1. In Deployer, go to the **Deployer > Projects** page.
2. Locate the project to which to authorize groups. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project. In the **Authorize** column for the project, click .
3. In the **Fetch Groups** field, click the type of group to authorize.

If you select **LDAP/Central user management groups**, you can narrow the list of groups that are displayed. In the **Search String** field, specify a string that is within the names of the groups you want to display; you can use an asterisk (*) as a wildcard for one or more characters (for example, Admin*). Click **Go** to list the specified groups.

4. In the **Select Authorization** list, click the task the group is authorized to perform.
5. The **Not Specified** box lists all groups defined in the type of group you chose. Using the arrow buttons, move each group you want to assign to the specified task into the **Allowed** box. Move each group that you do not want to assign to the specified task into the **Denied** box.
6. Click **Update**. The **Resulting users with this Authorization** lists all users that belong to the groups you assigned to the task (that is, the groups you moved into the **Allowed** box).
7. In the **Lock Status** column for the project, click  to unlock the project.

Adding and Viewing Instructions or Notes About a Project

When you create a project, Deployer automatically creates an HTML home page for the project. The HTML home page for a project is located in the *Integration Server_directory \ packages \ WmDeployer \ pub \ projects \ project* directory. The file name for the home page is *project.html*. Modify the page as necessary, but do not move it from this directory or rename it.

To view the home page for the project, go to the **Deployer > Projects** page and click  in the **Home** column for the project.

Update the Settings for a Project

You can update the settings for a specific project.

➤ To update the settings for a project

1. In Deployer, go to the **Deployer > Projects** page.


2. In the **Name** column of the **Projects** area, click the name of the project for which you want to update settings.
3. Update the settings as required and click **Save**.

Deleting a Project

Use the following procedure to delete a project you no longer need. When you delete a project, Deployer also deletes the deployment and deletion sets, builds, deployment maps, and deployment candidates associated with that project.

Deployer does not delete source repositories or composites used for repository-based deployment or source and target server aliases when you delete the project.

➤ To delete a project

1. In Deployer, go to the **Deployer > Projects** page.
2. Click  in the **Delete** column for the project.
Deployer displays a confirmation dialog.
3. Click **OK** to confirm that you want to delete the project.

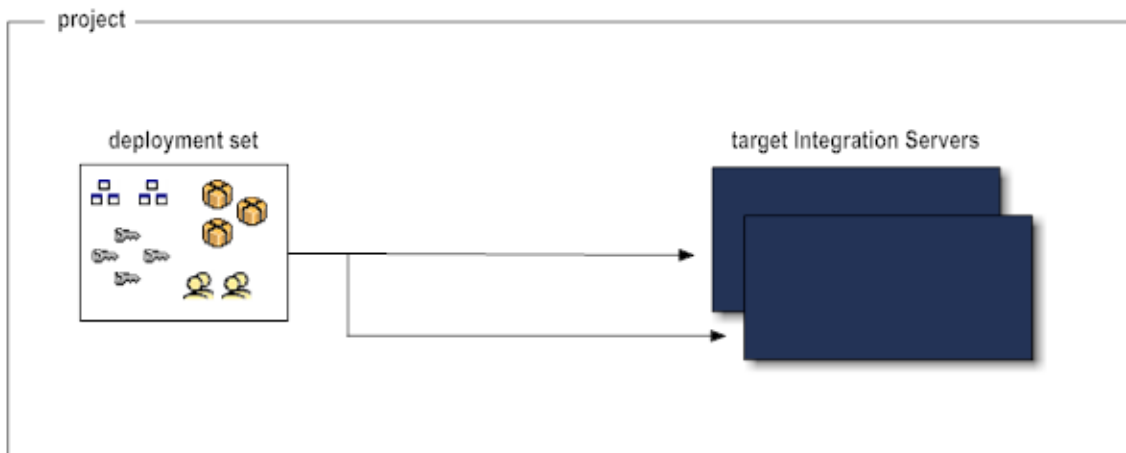
7 Mapping a Deployment Project to Target Servers

■ About Mapping a Project	218
■ Creating Target Groups	219
■ Mapping a Project to Target Servers and Target Groups	225
■ Mapping a Project with a Deletion Set for Runtime-based Deployment	227
■ Troubleshooting the Deployment Map	227
■ Exporting and Importing a Map	228
■ Substituting Configuration Values	229
■ Exporting and Importing Substitute Configuration Values	231

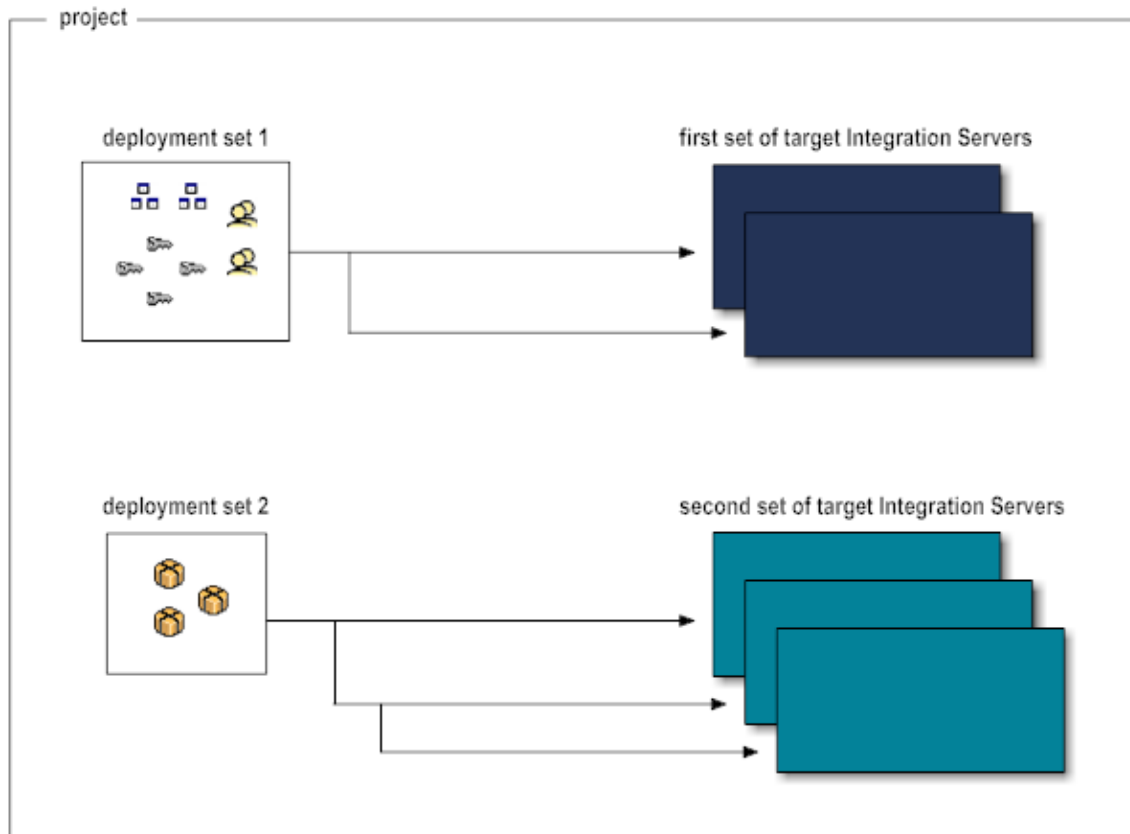
About Mapping a Project

In a *deployment map*, you identify target servers and target groups for each deployment set in a project. You can create multiple deployment maps for each project build (for example, if you are deploying to multiple environments).

To control to which targets, the assets are deployed, you define target servers. You define target servers according to the kind of assets you want to deploy. For example, if you want to deploy Integration Server assets to one set of target Integration Servers, you can define a single deployment set that identifies those assets.



To deploy some Integration Server assets to one set of target Integration Servers and other Integration Server assets to a second set of target Integration Servers, you must define two different deployment sets.



Creating Target Groups

If you find that you repeatedly have to map deployment sets to the same set of target servers, you can reduce your effort by grouping the target servers into a *target group*. You can then map the deployment sets to the target group rather than to the individual target servers.

Not all runtime types support the use of target groups when deploying to a clustered environment. The following table describes which runtimes do and do not support the use of target groups when deploying to a clustered environment:

Runtime	Use target groups when deploying to a clustered environment?
ActiveTransfer	No. ActiveTransfer groups (multiple instances sharing the same database), do not require target groups. A single ActiveTransfer Server instance in the group shares asset information with all other ActiveTransfer Server nodes. For more information, see <i>Managing File Transfers with webMethods ActiveTransfer</i> .
APIGateway	No. When APIGateway runs in clustered mode, you need to configure only one APIGateway in the cluster as a target. APIGateway takes care of the synchronization across all the cluster nodes.

Runtime	Use target groups when deploying to a clustered environment?
Application Platform	Yes. To deploy Application Platform assets to a clustered environment, you must set up connections to the cluster and create a target group that includes all of the servers in the cluster.
AgileApps Cloud	No
BAM process models	No.
BPM process models	Yes. For instructions on creating target groups for use in a clustered environment, see “Deploying to Clustered Integration Servers” on page 223.
Broker	Yes. To copy Broker clients to all of the cluster nodes, you must use a target group to deploy the Broker clients to each node in the cluster. When you deploy Broker assets to one of the nodes in a Broker cluster, all Broker assets except clients are copied to all of the cluster nodes.
Task Engine	Yes. To deploy task engine assets to a clustered environment, you should set up connections to the cluster and create a target group that includes all of the servers in the cluster.
Business Rules	Yes. To deploy business rules to a clustered environment, you should set up connections to the cluster and create a target group that includes all of the servers in the cluster.
Business Rules on My webMethods Server	No. Instead of using a target group containing all cluster nodes, add one target server only. Do not use the configured My webMethods Server Front End URL of the cluster, but the actual host and port of one of the configured cluster nodes. Make sure to configure this cluster node as described in Working with Business Rules in My webMethods. Since the deployed assets will be stored in the data base, all cluster nodes will be able to access them after deployment.
Digital Event Services	Yes. To deploy Digital Event Services (DES) assets to a clustered environment, you must configure connections to the cluster and create a target group that includes all of the servers in the cluster. The DES event types must be deployed to each single Software AG installation regardless of whether a server instance is part of a cluster.
EDA	Yes. To deploy EDA assets to a clustered environment, you must set up connections to the cluster and create a target group that includes all of the servers in the cluster.
Event Servers	Yes. You <i>must</i> use target groups to deploy to event servers in an HA cluster. You can also use a target group when deploying to multiple independent event servers.
	Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

Runtime	Use target groups when deploying to a clustered environment?
Integration Server	Yes. For instructions on creating target groups for use in a clustered environment, see “Deploying to Clustered Integration Servers” on page 223.
My webMethods	Yes. To deploy My webMethods assets to a clustered environment, you should set up connections to the cluster and create a target group that includes all of the servers in the cluster.
Optimize	No. When deploying Optimize assets to an Optimize cluster, you should deploy to a single node of that cluster. Do not deploy to a target group.
Trading Networks	Yes. For instructions on creating target groups for use in a clustered environment, see “Deploying to Clustered Integration Servers” on page 223.
Universal Messaging	Yes. You <i>must</i> use target groups to deploy cluster wide assets. For all other assets, you can use target groups if you want to sync the asset properties on each node.

➤ To create a target group

1. In Deployer, go to the **Target Groups > server** page.
2. Click **Create server Groups**.
3. In the **Name** field, type the name to use for the target group. The name can be up to 32 characters long and cannot contain spaces or the following special characters:

`$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "`

4. In the **Description** field, type a description for the target group. The description length has no limit and can include any characters.
5. In the **Version** box, enter the version of the target group.


Note:


Deployer limits the servers you can select for inclusion in the target group to those with the version you specify in the **Version** box. You cannot add servers of different versions to a target group. For information about selecting the version, see .


6. Click **Create**.
7. You can specify that deployment must either succeed on all servers in the target group or be automatically rolled back. In other words, if deployment fails on any server in the target group, you can specify that Deployer must automatically roll back the deployment on all servers in the group. To do so, set **Rollback All on Failure** to **Yes**; Deployer will ignore the **Rollback on Error** project setting (see [“Settings for Runtime-Based Deployment Projects”](#) on page 202).

Note:

Rollback All on Failure is valid for runtime-based deployment only. Deployer ignores this setting for repository-based deployment.

8. The **Available Servers** list shows the servers of the specified type for which you have set up connections to Deployer and that match the version you specified in the **Version** field. Select the servers to add to the target group, and then click **Add**. The servers move to the **Selected Servers** list.
9. Click **Save**.
10. To test the connection between Deployer and the target group, click  in the **Test** column in the left pane.

If the test fails, Deployer displays  **Resolve** in the **Test** column. You must resolve the servers to continue. Perform the following task to resolve the servers within the target group:


- a. Click  **Resolve** in the **Test** column.

Deployer displays the unresolved servers in the Check Inconsistencies page in the right pane.

- b. Click the server to resolve and click **Resolve Inconsistencies**.

Deployer removes the server from the target group and returns you to the Configure Target Group page.

- c. Make any additional changes to the target group and click **Save**.

- d. Click  in the **Test** column in the left pane to test the connection between Deployer and the target group.

11. If you want to rebuild the index, perform the following procedure:

Note:

For more information about rebuilding the index, see [“Rebuilding the Index” on page 69](#).


- a. If you want to change the repository directory, perform the following:

- a. In the **Name** column, click the name of the repository to edit.

Deployer opens the repository properties in the right-hand pane.

- b. In the **File Directory** box, type the full path of the repository for which to rebuild the index.

- c. Click **Save Changes**.

b. In the **Create Index** column for the repository, click .

For more information about rebuilding the index, see [“Rebuilding the Index” on page 69](#).

click  in the **Create Index** column on the left pane.

Deploying to Clustered Integration Servers

Deployer can deploy assets to clustered Integration Servers and to Trading Networks and process models running on Integration Servers. Keep the following points in mind when deploying to clustered Integration Servers:

- Before you can deploy to a cluster, you must define the connections to the Integration Servers in the cluster as remote servers and identify the Integration Servers as part of a target group in Deployer. You can then map and deploy to the target group as you would to any other target group. Since Trading Networks and process models run on clustered Integration Servers, to deploy Trading Networks assets and process models to a cluster, you must set up connections to the cluster and create a target group that includes the servers in the cluster in the same way.
- Before deploying Trading Networks assets to Trading Networks running on clustered Integration Servers, make sure the `tn.cluster.sync.remoteAliases` property is set for each Trading Networks server in the cluster. For instructions, see *webMethods Trading Networks Administrator's Guide*.
- To deploy process models to Process Engines running on clustered Integration Servers, you must configure each Integration Server hosting a process model in a cluster to use the same alias name and port number as the remote alias defined for the cluster.

Setting Up Connections to Integration Servers in the Cluster

➤ **To set up connections to the Integration Servers in the cluster**

1. Make sure all Integration Servers in the cluster are up and running.
2. In the Integration Server Administrator for the Integration Server that hosts Deployer, define every Integration Server in the Integration Server cluster as a remote server. For more information about remote servers, and instructions on defining them, see *webMethods Integration Server Administrator's Guide*.

Note:

All remote servers should have the same port number as the remote alias of the primary port of the cluster.

3. In Deployer, install the `WmDeployerResource` package on each Integration Server in the Integration Server cluster as follows:
 - a. Go to the **Servers > IS & TN** page; the page lists all Integration Servers you defined as remote servers.

- b. In the **Install** column, select the check box next to each Integration Server.
- c. Click **Install**.

Creating the Target Group

In a clustered environment, the only nodes from which you can select servers for the target group are those that are part of the primary configured port. To ensure that the servers are available for the target group, configure all servers in the target group to use the remote alias of the primary port.

Perform the following procedure to create a target group for a clustered environment.

> To create a target group in a clustered environment

1. In Deployer, perform one of the following:

For...	Perform the following...
Integration Server or Trading Networks	Go to Target Groups > IS & TN and click Create IS & TN Target Groups .
process models	Go to Create BPM(ProcessModel) Target Group and click Create BPM(ProcessModel) Target Group .

Note:

You should not use concurrent deployment when deploying BPM process models to BPM target groups.

2. In the **Name** box, type the name to use for the target group. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:

`$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "`

3. In the **Description** box, type a description for the target group. The description length has no limit and can include any characters.
4. In the **Version** box, enter the version of the target group.
5. Click **Create**.
6. In the left-hand pane, click the name of the target group from the **Group Name** column.
7. On the **Configure Target Group** pane, set **Roll Back All on Failure** to **Yes**.

Note:

Roll Back All on Failure is valid for runtime-based deployment only. Deployer ignores this setting for repository-based deployment.

- The **Available Servers** list shows the cluster name as a top-level node in the tree, and then all the servers in that cluster as child nodes under the cluster name. Select the cluster name node, and then click **Add**. The entire cluster tree moves to the **Selected Servers** list. Click **Save**.

Note:

If you select individual nodes of the cluster and not the entire cluster, when you deploy, the nodes in the cluster will no longer be identical. Tasks will not run equally well on all servers in the cluster, which could cause errors and failures.

If a server in the cluster is... Then...



If a server in the cluster is...	Then...
Defined as a remote server alias	The child node for that server shows the remote server alias, and you can select it.
Not defined as a remote server	The child node shows the server host and port, but you cannot select it.
Not running	That server does not appear in the list at all. Ensure that every server in the cluster is defined as a remote server and is up and running.

- You can add other clusters or individual servers to the target group.
- Map the project to the target group.
- Checkpoint, deploy, and, if necessary, roll back the project as you would in an unclustered environment.

Mapping a Project to Target Servers and Target Groups

You can map a project to individual target servers, target groups, or both.

> To map a project to target servers

- In Deployer, go to the **Deployer > Projects** page.
- If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
- In the **Name** column, click the project.
- In the right-hand pane, click  **Map**. Deployer displays the **Projects > project > Map** page and lists all maps that exist for the selected project.
- In the left-hand pane, click **Create Deployment Map**.

6. In the **Name** box, accept the default deployment map name or replace it with a name that you choose. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters:

\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "

7. In the **Description** box, type a description for the map. The description length has no limit and can include any characters.
8. Click **Create**.
9. Under the **Deployment Map Topology** area, in the **Set Mapping** column for a deployment set, click **Add Target Server** or **Add Target Group**. Based on the type of project you want to deploy, do one of the following:

- For runtime-based projects, select the check box next to each target server or group to which to deploy the assets in the deployment set and then click **Add**.

Deployer lists only those target servers or groups whose version is compatible with the source servers in the deployment set.

- For repository-based projects:

1. In the **Select Server** list, click the runtime type of the target server.
2. Select the check box next to each target server or group to which to deploy the assets in the deployment set and then click **Add**.

Repository-based deployment does not support mapping to target servers of versions that are different than the source repository. If you are mapping a repository-based project that contains BPM ProcessModels, you must also select the physical Integration Server servers or My webMethods Servers for each logical server in the deployment set from the **Map Logical Servers** area of the **Add Targets** pane.

If a server you want to map to does not appear in the list, you have not yet set it up to work with Deployer. After you connect Deployer to the target server you require, click **Refresh this Page** to update the list of servers on this page.

Note:

When you deploy Trading Networks assets, Deployer updates the Trading Networks database with the deployed assets. If Trading Networks is installed on multiple Integration Servers, map deployment sets that contain Trading Networks assets to only one of the Integration Servers. Do not map to multiple Integration Servers or you will experience unpredictable results when you deploy.



Mapping a Project with a Deletion Set for Runtime-based Deployment

If you are mapping a deletion set for runtime-based deployment, follow the same steps as described in [“Mapping a Project to Target Servers and Target Groups” on page 225](#). The only difference is that you must select target servers or groups from which to delete the assets.

Note:







When you deploy Trading Networks assets, Deployer updates the Trading Networks database with the deployed assets. If Trading Networks is installed on multiple Integration Servers, map deployment sets that contain Trading Networks assets to only one of the Integration Servers. Do not map to multiple Integration Servers or you will experience unpredictable results when you deploy.

Troubleshooting the Deployment Map

After mapping a project to target servers and groups, Deployer returns to the **map > Properties** page and the **Deployment Map Topology** area might show  or , which indicate if Deployer has encountered issues with the deployment map, as follows:

Note:

Keep the following points in mind when working with dependencies:

- Deployer cannot detect dependencies across products. Make sure assets you want to delete are not required by assets of other products.
- If you do not address problems at this time, Deployer will write messages about them to the simulation report. If you deploy without addressing problems, Deployer will not deploy the assets identified in the deployment set or delete the assets identified in the deletion set.
- For runtime-based deployment sets,  appears in the **Referenced Assets** column and indicates that you resolved an unresolved dependency using the **Exists** option, but Deployer has found that the referenced asset does *not* exist on target servers. Click  to see the missing referenced asset. You can then place the referenced asset on the target servers, or you can return to the project definition stage and re-resolve the dependency in a different way. For more information, see [“Resolving Dependencies” on page 45](#) (for runtime-based deployment).
- For repository-based deployment projects, Deployer verifies whether the target servers or target groups are available for deployment when you add them to the deployment map. The **Status** column shows  if the server is available for deployment and  if it is not.
- For deletion sets, dependencies work in the opposite direction from deployment sets. Deployer finds all assets on the target servers that depend on assets in the deletion set. If you were to delete the assets in the deletion set from the target servers, the dependent assets would no longer work properly. On the **map > Properties** page, in the **Deployment Map Topology** area, therefore, the  icon appears in the **Dependent Assets** column, and indicates that dependent assets exist. Click  to see the dependent assets, then choose whether to **Add** the dependent assets to the deletion set or to **Remove** the assets they depend on from the deletion set.


- If you are mapping a runtime-based deployment or deletion set and you resolved dependencies in the previous step, the contents of the deletion set have changed. As a result, you must rebuild the project (see [“Rebuilding a Build” on page 53](#)). If you exported the deletion set definition or the project build, you must also re-export the definition (see [“Exporting and Importing Deletion Set Definitions” on page 50](#)) and the build (see [“Exporting and Importing a Build” on page 53](#)).

Exporting and Importing a Map

You can export the deployment map from one Deployer environment and import to another Deployer environment. For example, you can export a map from the Deployer in your development environment to the Deployer in your testing environment. You must ensure that all target aliases in the test Deployer are the same as those in the development Deployer.

Before you import a map, you can edit any of the attributes (for example, you could map a deployment set to a different target server).

➤ To export and import a map

1. Export a map as follows:
 - a. In the source Deployer, go to the **Deployer > Projects > project > Map** page.
 - b. Locate the map to export and click  in the map's **Export** column. Deployer creates a file that contains the deployment map. The file is named *project_map.map* and is stored in the *Integration Server_directory \instances\instance_name\packages\WmDeployer\replicate\outbound* directory. Deployer also allows you to save the file to your local file system.
2. After you export a map, you can edit any of the attributes before importing it into the target environment. For example, you might want to map a deployment set to a new target server or target group. For instructions, see [“Editing a Deployment Map, Project Properties, or Substitute Configuration Values” on page 258](#).
3. Import the map as follows:
 - a. Copy the *project_map.map* file to the *Integration Server_directory \instances\instance_name\packages\WmDeployer\replicate\inbound* directory on the machine that hosts the target Deployer.
 - b. In the target Deployer, go to the **Deployer > Projects > project > Map** page.
 - c. Click **Import Map**, then select the *project_map.map* file you just copied to the inbound directory.

Substituting Configuration Values

Some assets might be configured differently on the source server (for runtime-based deployment) or repository (for repository-based deployment) than on the target server. You use Deployer to substitute different configuration values for assets during deployment so the assets will run properly on the target servers.

For example, as part of the deployment map for an IS & TN deployment set, you can specify configuration values for Integration Server assets that you want Deployer to substitute during deployment so the assets will run properly on target servers. Suppose an Integration Server in a development environment has a file polling port that is configured to monitor the C:\TEMP directory. You want to deploy this port to a production Integration Server on a Solaris system and have the port poll the /tmp directory instead. In the deployment map, you would specify a substitute configuration value of /tmp directory for the port. You can substitute different configuration values for scheduled tasks, ports, adapter connections, adapter notifications, and extended settings. You can substitute different configuration values for different target servers.

You can substitute configuration values as follows:

If you are creating a...	You can substitute configuration values by...
Runtime-based deployment project	Asset or target server. See “Substituting Configuration Values by Asset” on page 229 and “Substituting Configuration Values by Target Server (Runtime-Based)” on page 230.
Repository-based deployment project	Target server. See “Substituting Configuration Values by Target Server (Repository-Based)” on page 230.

Substituting Configuration Values by Asset

Note:

You can substitute configuration values by asset only in runtime-based deployment.

Perform the following steps to substitute configuration values by asset for runtime-based deployment projects.

➤ To substitute configuration values by asset

1. Under the **Deployment Map Properties** area, click **Configure Build by Assets**. Deployer lists assets that have configuration values in the left-hand pane.
2. Substitute different configuration values for an asset as follows:
 - a. In the left-hand pane, click the asset. Deployer displays the asset's configuration values as they exist on the source server.

- b. In the right-hand pane, type the configuration values to substitute.
- c. In the bottom right-hand pane, select the target servers or target groups on which to make the substitutions.
- d. Click **Save Substitutions**.

Substituting Configuration Values by Target Server (Runtime-Based)

Perform the following steps to substitute configuration values by target server for runtime-based deployment projects.


➤ To substitute configuration values by target server for a runtime-based project

1. Under the **Deployment Map Properties** area click **Configure Build by Servers**. Deployer lists the target servers that are mapped to the deployment set.
2. Select a target server. Deployer lists assets that have configuration values in the right-hand pane.
3. Substitute different configuration values for an asset as follows:
 - a. In the right-hand pane, click the asset. Deployer displays the asset's configuration values as they exist on the source server.
 - b. In the bottom right-hand pane, type the configuration values to substitute.
 - c. Click **Save Substitutions**.

Substituting Configuration Values by Target Server (Repository-Based)

Perform the following steps to substitute configuration values by target server for repository-based deployment projects.

➤ To substitute configuration values by target server for a repository-based project

1. On the **project > Map** page, in the **Configured** column, click .

Deployer opens a new page that displays target servers and target groups that are mapped to the deployment set.

2. In the **project > deployment map > Target Servers** pane, select the target servers for which you want to substitute values.

Deployer lists the composites that have configuration values in the center of the **Configurable Composites** pane.

3. Click the composite for which you want to substitute asset values.

Deployer displays the assets in the right-hand of the **Configurable Components** pane.

4. Click the assets in the **Name (Implementation Type)** column of the **Configurable Components** pane.

Note:

You can select more than one asset at a time.

Deployer displays the configuration values as they exist on the repository in the **Target Substitutions and Source Values** pane. If you selected multiple assets, Deployer displays the common properties but not source values.

Note:

Select the **Allow Empty Values** option in the **Target Substitutions and Source Values** pane if you want to save empty values for the properties.

5. In the bottom of the **Target Substitutions and Source Values** pane, type the configuration values to substitute.
6. Click **Save Substitutions** to save the substitutions, or **Restore Defaults** to clear the changes you made.

Note:

If the selected composites or the property value of the component is different across the selected servers, Deployer does not show any property value and the value field is enclosed by a red border. This occurs if multiple target servers are selected for variable substitution and the property value is different across different servers.

Exporting and Importing Substitute Configuration Values

You can export and import substitute configuration values for both runtime-based and repository-based deployment projects.

> To export and import substitute configuration values

1. Export the substitute configuration values from a deployment map as follows:
 - a. In the source Deployer, go to the **Deployer > Projects > project > Map** page.

- b. Click the deployment map that contains the substitute configuration values to export. Deployer displays the deployment map properties in the right-hand pane.
 - c. Click **Export Variable Substitution**. Deployer creates a file that contains the substitute configuration values for the assets in the project. The file is named *project_map.vs* and is stored in the *Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \outbound* directory. Deployer also allows you to save the file to your local file system.
 - d. If you exported substitute configuration values for scheduled tasks, open the *project_map.vs* file in an XML editor and set the task ID for each scheduled task to the task ID used on the target Integration Server.
2. Import the substitute configuration values into a deployment map as follows:
- a. Copy the *project_map.vs* file to the *Integration Server_directory \instances \instance_name \packages \ WmDeployer \replicate \inbound* directory on the machine that hosts the target Deployer.
 - b. In the target Deployer, go to the **Projects > project > Map** page.
 - c. Click the deployment map into which to import the substitute configuration values. Deployer displays the deployment map properties in the right-hand pane.
 - d. Click **Import Variable Substitution**.
 - e. Select the *project_map.vs* file you just copied to the inbound directory.

If you receive the error message "Input XML map information is not valid" while importing a variable substitution .vs file, open the file and do the following:

- a. Ensure that the project contains all deployment sets specified on the DeploymentSet nodes.
- b. For DeploymentSet nodes, ensure that the PluginGroup value is set to either true or false, and the PluginType value is correct.
- c. Ensure that each DeploymentSet node is mapped to the correct TargetSystem name as specified in the exported *project_map.vs* file
- d. Try to import again.

8 Deploying a Project

■	Generating a Checkpoint	234
■	Deploying a Project	234
■	Post-Deployment Tasks	237
■	Rolling Back Target Servers	238

Generating a Checkpoint

You generate a *checkpoint* for a project when you want the option of rolling back the target server to the state it was in prior to deploying your project. The checkpoint contains a copy of the assets on the target server that will be replaced by the assets in the deployment sets. You can set Deployer to generate checkpoints automatically or you can generate checkpoints manually for both runtime-based and repository-based projects.



Keep the following points in mind when working with checkpoints:

- If you take multiple checkpoints for a deployment candidate, only the latest is retained.
- The target servers must be available for the checkpoint generation to be successful.

When Deployer generates checkpoints automatically, it does so as the first step of the deployment process when you deploy a project. You set Deployer to create automatic checkpoints as part of creating the project. For runtime-based projects, you set the project to generate checkpoints automatically through the **Checkpoint Creation** parameter. For more information, see [“Settings for Runtime-Based Deployment Projects” on page 202](#). For repository-based projects, Deployer generates automatic checkpoints when you enable transactional deployment through the **Enable Transactional Deployment** parameter. For more information, see [“Creating a Project” on page 209](#).

You can generate a checkpoint manually for both runtime-based and repository-based projects.

➤ To generate a checkpoint manually

1. In the **Deployment Candidates** list, click  in the **Checkpoint** column. The checkpoint report appears in the right-hand pane in the **Deployment History** area.
2. Click  next to **Checkpoint** in the **Report Type** column to display the report. In the checkpoint report, the term **EXTRACT** is used for assets that exist on the target system and have been extracted to a backup. The term **MISSING** is used for assets that do not exist on the target system and will be deleted during a roll back.

The report is also available under the name `CheckpointReport_reportID.xml` in the `Integration Server_directory`
`\instances\instance_name\packages\WmDeployer\pub\projects\project_name\checkpoints\deployment_map\project_name\Checkpoint\reports` folder, where `project_name` is the name of the project and `deployment_map` is the name of the deployment map.



Deploying a Project

When you deploy a project, Deployer deploys the assets in the project to the target servers.

You can simulate a deployment before you actually deploy. When you simulate a deployment, Deployer generates a *simulation report* that scans the target servers and alerts you to some potential problems before you deploy. You can address problems and re-generate the simulation reports until all problems are resolved. A simulation report contains information such as the following:

- Assets that will be suspended during deployment.
- Assets that will be enabled after deployment.
- Changes that will occur on the target servers, such as the assets that will be added or overwritten, and configuration values that will be substituted for Integration Server assets.
- Messages about problems, such as unresolved dependencies.



> To deploy a project


1. If you chose to suspend triggers, ports, and scheduled tasks, but a service is triggered by one of these assets before Deployer suspends them, and the service is a long-running service, Deployer might overwrite the service during deployment. Make sure long-running services have completed.
2. In Deployer, go to the **Deployer > Projects** page.
3. If locking is enabled, in the **Lock Status** column for the project, click  to lock the project.
4. In the **Name** column, click the project.
5. In the right-hand pane, click  **Deploy**. Deployer displays the **Projects > project > Deploy** page and lists all deployment candidates that exist for the selected project.
6. In the left-hand pane, click **Create Deployment Candidate**.
7. Set the **Create Deployment Candidate** parameters as follows:

Parameter	Description
Name	Accept the default deployment candidate name or replace it with a name that you choose. The name can be up to 32 characters long and cannot contain spaces or the following illegal characters: \$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "
Description	Type a description for the deployment candidate. The description length has no limit and can include any characters.
Project Build	(Run-time based deployment only.) Click the project build to deploy.
Deployment Map	Click the deployment map that identifies the target servers to which to deploy the assets. If the words Missing referenced assets appear next to the map name in the list, it means that you resolved an unresolved dependency using the Exists option, but the referenced asset does not exist on the target server. You can place the referenced asset on the target servers, or you

Parameter	Description
	<p>can return to the project definition stage and re-resolve the dependency in a different way. For more information, see “Resolving Dependencies” on page 45 (for runtime-based deployment) or “Resolving Dependencies” on page 91 (for repository-based deployment).</p> <p>If you do not address the problem during the mapping task, Deployer will write a message about the problem to the simulation report. If you deploy without addressing the problem, Deployer will not deploy the deployment set.</p>

8. Click **Create**.


In the candidate list in the left-hand pane, if the selected build and the current project definition are in sync, the **Status** column shows . If the project definition has changed since the build was created, the column shows .


9. If you want to see the progress report, click  in the **Progress Report** column.

The progress report displays the updates for simulate, deploy, checkpoint and rollback requests as they occur. This is useful in the case where the deployment build is large and it takes a long time to complete the action.

Note:


If you are deploying a runtime-based project, you can rebuild the project build before proceeding. For instructions, see [“Rebuilding a Build” on page 53](#).


10. If you want to simulate the deployment, in the **Deployment Candidates** list, click  in the **Simulate** column.

The simulation report appears in the right-hand pane in the **Deployment History** area. Click  next to **Simulation** in the **Report Type** column to display the report. Read the report and address all problems. The report is also available under the name `project_name_previewReport_reportID.xml` in the `Integration Server_directory\instances\instance_name\packages\WmDeployer\pub\projects\project_name\targets\deployment_map\reports` folder, where `project_name` is the name of the project and `deployment_map` is the name of the deployment map.

Note:

If you do not address all problems at this time, you will probably experience errors during the deployment. For instructions on resolving unresolved dependencies, see [“Resolving Dependencies” on page 45](#) (for runtime-based deployment) or [“Resolving Dependencies” on page 91](#) (for repository-based deployment).

11. Click  in the **Deploy** column for the deployment candidate. Deployer does the following:

- Deploys the assets in the project to the target servers.
- Creates a deployment report and lists the report in the **Deployment History** area. Click  next to **Deployment Report** in the **Report Type** column to display the report. The report contains similar information to the simulation report, except that the events have actually occurred at this point. The report is also available under the name `project_name_auditReport_reportID.xml` in the *Integration Server_directory* `\instances\instance_name\packages\WmDeployer\pub\projects\project_name\targets\deployment_map\reports` folder, where `project_name` is the name of the project and `deployment_map` is the name of the deployment map.
- If you are creating a runtime-based deployment project, Deployer performs the following additional tasks:
 - If you chose automatic checkpointing or automatic rollback in the project properties, Deployer automatically generates a checkpoint at this time. If you chose manual checkpointing and no checkpoint exists, Deployer asks whether you want to deploy anyway. If you deploy without a checkpoint, you will not be able to roll back the target servers.
 - If the project build contains deletion set definitions, Deployer deletes the specified assets from the target servers you identified in the selected deployment map.
 - If you are creating a repository-based deployment project and you set the **Enable Transactional Deployment** property to **Yes**, Deployer creates the checkpoint for the target server. For more information about the **Enable Transactional Deployment** property, see [“Creating a Project” on page 209](#).

Post-Deployment Tasks

- If you deployed JMS triggers, do the following:
 1. Create the same JMS alias connections on the target Integration Servers that exist on the source Integration Servers. Then reload the packages that contain the triggers.
 2. Enable the JMS triggers.
 3. Configure the queue or topic for each JMS trigger on the message provider for the target Integration Servers.

For instructions, see *Using webMethods Integration Server to Build a Client for JMS*.

- If you deployed My webMethods Server rules, the order in which the deployed rules are resolved with the existing rules on the target servers might need modification. Review the rule order and modify as necessary.
- If you deployed a process that uses e-forms with the project property **Enable process for execution** set to **No** (see [“Creating a Project” on page 209](#)), the e-form listener associated with the process cannot be enabled on the target server and therefore the process cannot be triggered on the target server. To make the process triggerable on the target server, enable it for execution and then enable the e-form listener.

Rolling Back Target Servers

If deployment to a target server fails and the target environment is in an inconsistent state, or a deployment is successful but the deployed assets are not working as expected, you can use Deployer's roll back feature to undo the deployment. When you roll back a deployment, Deployer rolls back the target server to the last checkpoint generated for the project. For more information about generating checkpoints, see [“Generating a Checkpoint” on page 234](#).

You can set Deployer to roll back target servers automatically or you can roll back target servers manually after deployment.

Deployer automatically rolls back target servers for runtime-based projects in these cases:

- You set the **Rollback on Error** project setting to **Automatic** (see [“Settings for Runtime-Based Deployment Projects” on page 202](#)). If the deployment fails on a target server, Deployer automatically rolls back that target server.
- Deployment failed to a target group whose **Rollback All on Failure** setting is **Yes** (see [“Creating Target Groups” on page 219](#)). If deployment to any server in such a target group fails, Deployer automatically rolls back all servers in the target group.

Deployer automatically rolls back the target servers for repository-based projects when transactional deployment is enabled for the project and deployment fails. For more information about enabling transactional deployment, see [“Creating a Project” on page 209](#).


For runtime-based projects, you can roll back target servers manually at any time after deployment if you performed *both* of the following:

- You set the **Rollback on Error** project setting to **Manual**.
- You did not deploy to a target group whose **Rollback All on Failure** setting is **Yes**.


For repository-based projects, you can roll back target servers manually if you performed *either* of the following:

- You enabled transactional deployment to create an automatic checkpoint through the **Enable Transactional Deployment** parameter.
- You generated a manual checkpoint for the project.

➤ To roll back target servers manually

1. In the **Deployment Candidates** list, click  in the **Rollback** column.

Deployer displays the rollback report in the right-hand pane in the **Deployment History** area.

2. To display the rollback report, click  next to **Rollback** in the **Report Type** column. The report is also available under the name `project_name_auditReport_reportID.xml` in the `Integration Server_directory \instances \instance_name \packages \WmDeployer \pub \projects \project_name \targets \deployment_map \reports` folder, where

project_name is the name of the project and *deployment_map* is the name of the deployment map.

If you rolled back an IS & TN deployment set, the following apply:

- If the **Activate After Deployment** option for a package was set to **Inbound Only**, the report will warn that the package is not present on the target Integration Servers. You can ignore this warning.
- If the deployment set included webMethods files, the directory structure for those files remains in the webMethods installation directory on the target servers. You can delete the directories manually.
- If you deployed Trading Networks document attributes, field definitions, binary types, or profile security data, Deployer does not roll them back.

If you rolled back a ProcessModel deployment set, the rollback behavior varies, as follows:

- For process models you deployed that were versions of existing process models on the target servers, Deployer rolled back the deployed versions from the target servers.
- For deployed process models that were new on the target servers, Deployer disables the deployed process models but does not remove them from the target servers. However, Deployer removes the Integration Server assets from the target servers on rolling back a project with deployment sets that include IS assets for the process models. If a process instance for the rolled-back process model exists on a target server, this behavior could result in issues.

If you rolled back Universal Messaging assets, the rollback behavior is as follows:

- Deployer will not roll back the assets if the port is already being used by another existing interface.
- Deployer does not roll back nested security groups. For example, if group1 contains group2, the rollback will not restore the nested group2.

9 Using Deployer Commands

■ Installing Command Line Interface Only	242
■ Creating and Running Scripts	242
■ Specifying Logon Parameters	245
■ Error Handling and Logging	248
■ General and Project Commands	248
■ Build Commands	252
■ Commands for Repository-Based Deployment	257
■ Map Commands	257
■ Deployment Commands	261

Installing Command Line Interface Only

Typically, you install the Deployer command line interface when installing Deployer from the Software AG Installer. If you use an automated deployment procedure that spans multiple machines, you can install the Deployer command line interface on a machine, without installing the rest of Deployer or a host Integration Server.

To install only the Deployer command line interface on a machine, open the Deployer.{bat|sh} scripts for an existing remote Deployer host machine and copy all jars or folders specified in the `-classpath` parameter to the same location on the local machine. In the Deployer.{bat|sh} file you copied to the local machine, make sure that the `-classpath` parameter points to the jar files you copied and to a JDK or JRE on the local machine.

Creating and Running Scripts

You can enter Deployer commands at a command prompt or you can create scripts that execute commands automatically. If you create a script, Deployer runs the commands in the order in which they appear in the script.

To invoke Deployer from the command line and execute a script, use the command for your operating system as follows:

For...	Command
Windows or UNIX	Deployer.{bat sh} <i>path_to_file</i>
Mac	deployerMac.sh <i>path_to_file</i>

You can also call scripts from other automated procedures, such as other scripts.

The following sample script automates these tasks on a Windows system:

- Imports a build that was exported from a test environment. Deployer automatically creates the deployment project and deployment sets.
- Displays the build contents on the console.
- Imports the deployment map.
- Imports substitute configuration values for Integration Server assets into the deployment map.
- Creates a deployment candidate.
- Generates a checkpoint, simulates the deployment, and deploys the build.

```
:environment
set host=%1
set port=%2
set user=%3
set pwd=%4
set project=testProject
set build=DemoBuild
set depCandidate=DemoDC
```

```

set depMap=DemoMap
rem ----clear the ERRORLEVEL system variable to avoid any side effects of
previous executions cases
set ERRORLEVEL=
:importBuild
set importB=%project%_ExportedBuild_%build%
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Importing Build %ImportB%
ECHO -----
call Deployer.bat --import -buildFile %importB% -host %host% -port %port% -user
%user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set importB=
set nextAction=describeBuild
GOTO verifyStatus
:describeBuild
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Describing %build%
ECHO -----
call Deployer.bat --describe -build %build% -project %project% -host %host% -port
%port% -user %user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set nextAction=buildit
GOTO verifyStatus
:importMap
set importM=%project%_%depMap%.map
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Importing Map %ImportM%
ECHO -----
call Deployer.bat --import -mapFile %importM% -project %project% -host %host%
-port
%port% -user %user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set importM=
set nextAction=importVarSub
GOTO verifyStatus

:importVarSub
set importV=%project%_%depMap%.vs
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Importing Varsub %ImportV%
ECHO -----
call Deployer.bat --import -varsub -vsFile %importV% -map %depMap% -project
%project% -host %host% -port %port% -user %user% -pwd %pwd%
@echo off
echo.

```

```

echo.
echo.
set importV=
set nextAction=createDC
GOTO verifyStatus

:createDC
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Creating Deployment Candidate %depCandidate%
ECHO -----
call Deployer.bat --create -dc %depCandidate% -build %build% -map %depMap%
-project
%project% -host %host% -port %port% -user %user% -pwd %pwd%
@echo off
echo.
echo.
echo.
set nextAction=simulate
GOTO verifyStatus
:simulate
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Performaing deployment simulation on deployment candidate %depCandidate%
ECHO -----
call Deployer.bat -host %host% -port %port% -user %user% -pwd %pwd%
--simulate -project %project% -dc %depCandidate%
@echo off
echo.
echo.
echo.
set nextAction=checkpoint
GOTO verifyStatus
:checkpoint
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO Performing CHECKPOINT operation of %depCandidate%
ECHO -----
echo %project%
echo %depCandidate%

call Deployer.bat --checkpoint -project %project% -dc %depCandidate% -host %host%
-port %port% -user %user% -pwd %pwd%
@echo off
echo .
echo .
echo .
set nextAction=deploy
GOTO verifyStatus
:deploy
IF "%ERRORLEVEL%" == "8" GOTO FINISH
ECHO -----
ECHO DEPLOYING %depCandidate%

:VerifyStatus
IF "%ERRORLEVEL%" == "8" ECHO "<<<ERROR>>>"
IF "%ERRORLEVEL%" == "4" ECHO "<<<WARNING>>>"
IF "%ERRORLEVEL%" == "0" ECHO "<<<SUCCESS>>>"
echo.
echo.

```

```
goto %nextAction%
:FINISH
echo.
echo.
echo Completed.
set host=
set port=
set user=
set pwd=
set project=
set build=
set depCandidate=
set ERRORLEVEL=
@echo on
```

Specifying Logon Parameters

All Deployer commands require parameters for logging into Integration Server that hosts Deployer. You can have Deployer commands connect to Integration Server using HTTP or HTTPS.

If you want Deployer commands to log on using HTTP, you can use an existing HTTP port on Integration Server or configure a new one.

If you want Deployer commands to log on using HTTPS, do the following:

- Use an existing HTTPS port on Integration Server or configure a new one.
- Place the command line interface's client certificate, private key, and signing authority's certificate on the Integration Server host machine.
- Map the command line interface's client certificate to an Integration Server user that has Administrator or Developer privileges.

For instructions for these tasks, see the *webMethods Integration Server Administrator's Guide*.

When you run Deployer commands, the logon parameters you provide depend on whether you want to use HTTP or HTTPS.

■ Logon parameters for logging into an HTTP port

```
Deployer.{sh|bat} --command -host host -port port-user user-pwd password
```

■ Logon parameters for logging into an HTTPS port

```
Deployer.{sh|bat} --command -host host -port port-user user-pwd password
-useSSL -senderCert path_to_cert-privKey path_to_key-caCert path_to_cert
```

Ensure that the certificates are in DER format; if not, convert them to DER format using a certificate management tool, such as Java keytool. See the *Prerequisites to Configuring a Port for SSL* section in the *webMethods Integration Server Administrator's Guide* for more information on the type of certificate format supported.

The following table shows the parameters required when you log into an HTTPS port with respect to different options in the **Security Configuration** section in Integration Server Administrator.

(Integration Server Administrator > Ports > Add Port > Select Type of Port to Configure as webMethods/HTTPS > Security Configuration > Client authentication section.

If you select the Client Authentication option as...

Provide the following parameters...

Username/Password

Mandatory parameters

- Username (-user)
- password (-pwd)

Optional parameters

- -senderCert
- -privKey
- -caCert

Request Client Certificates

Mandatory parameters

- Username (-user)
- password (-pwd)

or provide

Mandatory parameters

- -senderCert
- -privKey

Optional parameters

- -caCert

Require Client Certificates

Mandatory parameters

- -senderCert
- -privKey

Optional parameters

- -caCert

If you select Java Secure Socket Extension (JSSE) in the **Security Configuration** section in Integration Server Administrator (**Integration Server Administrator > Ports > Add Port > Select Type of Port to Configure as webMethods/HTTPS > Security Configuration > Use JSSE > Select Yes**), the logon parameters for logging into an HTTPS port are as follows:

```
Deployer.{sh|bat} --command -host host -port port-user user-pwd password
-useSSL -senderCert path_to_cert-privKey path_to_key-caCert path_to_cert -useJSSE
```

Ensure that the certificates are in DER format; if not, convert them to DER format using a certificate management tool, such as Java keytool. See the *Prerequisites to Configuring a Port for SSL* section in the *webMethods Integration Server Administrator's Guide* for more information on the type of certificate format supported.

Parameter	Description
<code>-host host -port port</code>	Integration Server logs on to this host machine and port.
<code>-user user -pwd password</code>	User name and password to use to log on to Integration Server. Note: If you do not provide a password, Deployer prompts you for the password.
<code>-useSSL</code>	Log on to an HTTPS port.
<code>-senderCert path_to_cert</code>	Command line interface's client certificate.
<code>-privKey path_to_key</code>	Command line interface's private key.
<code>-caCert path_to_cert</code>	Command line interface's signing authority's certificate. If the certificates and private key do not exactly match the ones in the Integration Server installation for the command line interface, the command will fail.
<code>-useJSSE</code>	Log on to an HTTPS port with Java Secure Socket Extension (JSSE) enabled. If an HTTPS port makes use of JSSE, then pass <code>useJSSE</code> along with the <code>useSSL</code> option.

Creating a Configuration File for Logon Parameters

You can save time by creating a configuration file that specifies the values to use for the logon parameters and then pointing commands to the configuration file. Create the configuration file using a text editor and specify the appropriate parameter values. For example:

```
host=idcauto1
port=5555
user=Administrator
pwd=1xcfdg55
```

```
host=idcauto1
port=5555
useSSL=true
senderCert=C:/files/SenderCert.der
```

```
privKey=C:/files/SenderPrivKey.der  
caCert=C:/files/SenderCACert.der
```

Save the file with the extension `.cnf` and store it in the *Integration Server_directory* \instances\instance_name\packages\WmDeployer\bin directory.

To point a command to the configuration file, specify the following on the command instead of the logon parameters:

```
Deployer.{sh|bat} --command -configfile file
```

Parameter	Description
<code>command</code>	Command to run.
<code>-configfile file</code>	Full path to the configuration file.

Error Handling and Logging

Deployer logs errors that occur during command line operations in the Deployer command line log file. The log file is named `CLI.log` and is located in a directory inside the current working directory. For example, if your working directory is *Integration Server_directory* \instances\instance_name\packages\WmDeployer, `CLI.log` is located in the *Integration Server_directory* \instances\instance_name\packages\WmDeployer\logs directory.

Typical command line errors include required options that were not specified and invalid parameter values. Execution errors can include connectivity and authentication errors.

The maximum size for the `CLI.log` file is 100 KB. When it reaches the maximum size, it archives the log by renaming the file `CLI.log.old` and creating a new `CLI.log` file.

General and Project Commands

This section describes the commands to display Deployer usage information and product details and to maintain projects.

About Command

The `--about` command displays the following details about Deployer:

- JVM version number
- Publisher information
- Build number
- Package name
- Copyright information
- Integration Server version number

Run the following command to see the project details:

```
Deployer.{sh|bat} --about -host host -port port -user user_name -pwd password
```

Deleting a Project

Run the following command to delete a project:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --delete -project project  
-host host -port port -user user_name -pwd password
```

For more information about deleting projects, see [“Deleting a Project” on page 216](#).

Displaying Project Properties

Run the following command to display project properties:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --getProjectProperties -project project  
-host host -port port -user user_name -pwd password
```

Exporting Deletion Sets from a Project

When you run the `--export` command, Deployer creates a file that contains the definitions. The file is named `project_deleteSets.xml` and is stored in the `Integration Server_directory \instances\instance_name\packages\WmDeployer\replicate\outbound` directory.

Run the following command to export a deletion set:

Note:

You must have Define ACL authorization to run this command.

```
Deployer.[sh|bat] --export -deleteSpec -project project  
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-project <i>project</i></code>	Project whose deletion set definitions to export.
<code>-overwrite</code>	If the project already contains a file with the same name, this option tells Deployer to overwrite it. If you do not overwrite, and a file with the same name exists, Deployer issues an error and ends the command.

Importing Deletion Set Definitions into a Project

Before you can import deletion set definitions, you must copy the exported *project_deleteSets.xml* file to the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\inbound directory.

If the project already contains a deletion set with the same name as one you are importing, Deployer issues an error and ends the command.

Run the following command to import a deletion set into a project:

Note:

You must have Define ACL authorization to run this command.

```
Deployer.{sh|bat} --import -deleteSpec definitions_file -project project
-host host-port port-user user_name -pwd password
```

Parameter	Description
-deleteSpec <i>definitions_file</i>	Full path to the file that contains the definitions to import. Definition files are named <i>project_deleteSets.xml</i> and are located in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmDeployer\replicate\inbound directory.
-project <i>project</i>	Project into which to import the definitions.

Exporting Project Properties

When you export a project's properties, Deployer creates a file that contains the project property settings. The file is named *project.properties* and is stored in the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\outbound directory. For more information about exporting project properties, see [“Exporting and Importing Project Properties” on page 212](#).

Run the following command to export project properties:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --export -projectProperties project
-host host -port port -user user_name -pwd password
```

Parameter	Description
projectProperties <i>project</i>	Project from which to export properties.

Importing Project Properties

Importing properties into a project overwrites the existing properties for that project.

Before you can import project properties, you must copy the exported *project.properties* file to the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\inbound directory on the machine that hosts the target Deployer.

You can edit the properties before you import them (see [“Editing a Deployment Map, Project Properties, or Substitute Configuration Values”](#) on page 258). If you do, keep in mind the following:

- You can specify ALWAYS or NEVER for the `overwrite` property.
- You can specify REPLACE OR MERGE for the `deployTNRules` property.
- You can specify true, false, or selected for the `stopTriggers` property.
- You can specify true or false for all other properties.

If you specify a value for a property that is not allowed, Deployer resets the property to the default value when it imports the project properties.

Run the following command to import a project’s properties:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --setProjectProperties -project project
-projectFile properties_file
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-project <i>project</i></code>	Project into which to import the properties.
<code>-projectFile <i>properties_file</i></code>	Full path to the file that contains the properties to import. These files are named <i>project.properties</i> and are located in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmDeployer\replicate\inbound directory.

Help

Run the following command to view a list of Deployer commands you can use in the command line interface:

```
Deployer.{sh|bat} --help -command command_string
```

Parameter	Description
<code>-command <i>command_string</i></code>	Command for which you want usage information.

Listing Builds, Maps, or Deployment Candidates for a Project

Run the following command to list builds, maps, or deployment candidates for a build:

Note:

You must have the correct authorizations to run this command depending on whether you want to list builds, maps, or deployment candidates.

- To list builds, you must have Build ACL authorization.
- To list maps, you must have Map ACL authorization.
- To list deployment candidates, you must have Deploy ACL authorization.

```
Deployer.{sh|bat} --list -candidate {Build|Map|DC} -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
-candidate {Build Map DC}	Whether to list builds, maps, or deployment candidates.
-project <i>project</i>	Project that contains the builds, maps, or deployment candidates to list.

Locking Projects

Run the following command to lock a project:

```
Deployer.{sh|bat} --lockProject -project project
-host host -port port -user user_name -pwd password
```

Unlocking Projects

Run the following command to unlock a project:

```
Deployer.{sh|bat} --unlockProject -project project
-host host -port port -user user_name -pwd password
```

Build Commands

This section describes the commands to create, export, import, and display details about a build.

Creating a Project Build

When creating a project build, the build creation will fail if there are any unresolved dependencies. For instructions on resolving unresolved dependencies, see [“Resolving Dependencies” on page 45](#).

Run the following command to create a project build:

Note:

You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --create -build build -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
-build <i>build</i>	Name of the build to create. The build name can be up to 32 characters long and can include any characters that are valid for a file name in your operating system.
-project <i>project</i>	Project from which to create the build.
-reportFilePath <i>report_path</i>	Full path to the local directory where Deployer stores the generated build report.

Listing Builds for a Project

Run the following command to list the builds in a project:

Note:

You must have View ACL authorization to run this command.

```
Deployer.{sh|bat} --list -candidate build -project project
```

```
-host host -port port -user user_name -pwd password
```

Displaying Contents of a Build

Run the following command to display the contents of a specific build:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -build build -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
-build <i>build</i>	Build whose contents to display.
-project <i>project</i>	Project to which the build belongs.

Displaying Substitute Configuration Values for Integration Server Assets in a Build

Use the following command to display the substitute configuration values for Integration Server assets in a build:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -build build -project project-varsub
-host host -port port -user user_name -pwd password
```

Parameter	Description
-build <i>build</i>	Build whose substitute configuration values to display.
-project <i>project</i>	Project to which the build belongs.
-varsub	Displays the substitute configuration values.

Displaying Contents of a Build File

Use the following command to display the contents of a build file:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -buildFile build_file -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
-buildFile <i>build_file</i>	Full path to the build file whose contents to display. Build files are named <i>project_build</i> and are located in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmDeployer\replicate\outbound directory.
-project <i>project</i>	Project to which the build belongs.

Displaying Substitute Configuration Values for Integration Server Assets in a Build File

Use the following command to display the substitute configuration values for Integration Server assets in a build file:

Note:

You must have Administrator ACL authorization to run this command.

```
Deployer.{sh|bat} --describe -buildFile build_file -project project-varsub
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-buildFile <i>build_file</i></code>	Full path to the build file whose substitute configuration values to display. Build files are named <i>project_build</i> and are located in the <i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmDeployer\replicate\outbound directory.
<code>-project <i>project</i></code>	Project to which the build belongs.
<code>-varsub</code>	Displays the substitute configuration values.

Exporting a Build from a Project

Use the following command to export a build from a project:

Note:

You must have Build ACL authorization to run this command.

```
Deployer.[sh|bat] --export -build build -project project
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-build <i>build</i></code>	Build to export.
<code>-project <i>project</i></code>	Project to which the build belongs.
<code>-overwrite</code>	If the project already contains a build with the same name, this options tells Deployer to overwrite it. If you do not overwrite, and a build with the same name exists, Deployer issues an error and ends the command.

Deployer creates a file that contains the build. The file is named *project_build* and is stored in the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\outbound directory.

Importing a Build File into a Project

Before you can import a build, you must copy the exported *project_build* file to the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\inbound directory on the machine that hosts the target Deployer.

Run the following command to import a build file into a project.

Note:

You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --import -buildFile build_file-project project
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-buildFile <i>build_file</i></code>	Full path to the build file that contains the deployment map to import. Build files are named <i>project_build</i> and are located in the <i>Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \inbound</i> directory.
<code>-project <i>project</i></code>	Project into which to import the build.
<code>-overwrite</code>	If the project already contains a build with the same name, this options tells Deployer to overwrite it. If you do not overwrite, and a build with the same name exists, Deployer issues an error and ends the command.

Listing Build Reports

Run the following command to list the build reports for a project.

Note:

You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --list -candidate buildReport -build build -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-build <i>build</i></code>	Build for which to list build reports.
<code>-project <i>project</i></code>	Project to which the build belongs.

Displaying a Build Report

Run the following command to display a build report.

Note:

You must have Build ACL authorization to run this command.

```
Deployer.{sh|bat} --showReport -candidate buildReport -build build
-id integerId -project project -host host -port port -user user_name
-pwd password
```

Parameter	Description
<code>{-build <i>build</i></code>	Build whose build report to display.
<code>id <i>report_</i> <i>identifier</i></code>	Identifier for the report to display. Use the <code>--list</code> command (see “Listing Build Reports” on page 256) to display report identifiers, as well as the date and time each report was generated.

Parameter	Description
<code>-project project</code>	Project to which the build belongs.

Commands for Repository-Based Deployment

This section describes commands you can run specific to building indexes for repository-based deployment.

Rebuilding the Index with the Build Script

If the index you created becomes corrupted or the repository index is accidentally deleted from the repository, you can use the `createIndex` command to recreate the index.

By default, the `createIndex` command rebuilds the index in the location specified by the `build.output.dir` property you specified in [“Setting Build Properties” on page 58](#). You can override the default repository path by specifying the path of the repository with the `-Drepo.dir` command.

Note:

When you follow this procedure to rebuild the index, the build script creates *only* the index. To build the index, check out the asset sources, version the assets, and build the composites and descriptors in the repository, you must run the build script as described in [“Running the Build Script” on page 65](#).

Run one of the following commands from the *Software AG_directory* \common\AssetBuildEnvironment\bin directory:

For this platform...	Run the following command...
Windows	<code>build.bat -Drepo.dir=repository_path createIndex</code>
UNIX	<code>build.sh -Drepo.dir= repository_path createIndex</code>

Where *repository_path* is the full path of the repository directory.

Note:

If you do not specify a path for `-Drepo.dir`, the build script indexes the repository specified by the `build.output.dir` property.

Map Commands

This section describes the commands to list, import, export, edit, and delete deployment maps.

Note:

You must have Map ACL authorization to run the commands in this section.

Listing All Deployment Maps

Run the following command to list the deployment maps for a candidate.

```
Deployer.{sh|bat} --list -candidate mapFile
-host host -port port -user user_name -pwd password
```

Exporting a Deployment Map from a Project

When you export a deployment map from a project, Deployer creates a file that contains the deployment map. The file is named *project_map.map* and is stored in the *Integration Server_directory* \instances\instance_name\packages\WmDeployer\replicate\outbound directory.

Run the following command to export a deployment map:

```
Deployer.{sh|bat} --export -map map -project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
-map <i>map</i>	Deployment map to export.
-project <i>project</i>	Project to which the map belongs.

Editing a Deployment Map, Project Properties, or Substitute Configuration Values

After you export a deployment map or substitute configuration values, you can edit the resulting file before importing it into the other environment. For example, if you want to map a deployment set to a different target server, you could change the `targetServer alias` attribute to reflect the new target server name.

You can open a deployment map or substitute configuration value file using any XML editor. A deployment map file has the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeploymentMap description="description of map" mapName="mapSetName"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DeploymentSets>
    <DeploymentSet name="deploymentSetName" pluginType="pluginType">
      <targetGroups>
        <targetGroup alias="targetGroupName"/>
      </targetGroups>
      <targetServers>
        <targetServer alias="targetServerAlias"/>
      </targetServers>
    </DeploymentSet>
  </DeploymentSets>
</DeploymentMap>
```

To specify an additional target server, target group, or deployment set in the same deployment map, repeat the attribute for each addition. For example, a deployment set that is mapped to multiple target servers is defined as follows:

```
<DeploymentMap>
  <DeploymentSets>
    <DeploymentSet name="deploymentsetA" pluginType="MWS">
      <targetGroups>
        <targetGroup alias="<targetGroupName>"/>
      </targetGroups>
      <targetServers>
        <targetServer alias="server1"/>
        <targetServer alias="server2"/>
        <targetServer alias="server3"/>
      </targetServers>
    </DeploymentSet>
  </DeploymentSets>
</DeploymentMap>
```

Importing a Deployment Map Into a Project

Before you can import a deployment map, you must copy the exported *project_map.map* file to the *Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \inbound* directory on the machine that hosts the target Deployer. You can edit the map before you import it (see [“Editing a Deployment Map, Project Properties, or Substitute Configuration Values” on page 258](#)).

Run the following command to import a deployment map into a project:

```
Deployer.{sh|bat} --import -mapFile map_file -project project
-overwrite -host host -port port -user user_name -pwd password
```

Parameter	Description
-mapFile <i>map_file</i>	Full path to the map file that contains the deployment map to import. Map files are named <i>project_map.map</i> and are located in the <i>Integration Server_directory \instances \instance_name \packages \WmDeployer \replicate \inbound</i> directory.
-project <i>project</i>	Project into which to import the map.
-overwrite	If the project already contains a map with the same name, this options tells Deployer to overwrite it. If you do not overwrite, and a map with the same name exists, Deployer issues an error and ends the command.

Exporting Substitute Configuration Values for Integration Server Assets from a Deployment Map

Run the following command to substitute configuration values for Integration Server assets from a deployment map:

```
Deployer.{sh|bat} --export -map map -project project -varsub
```

```
-host host -port port -user user_name -pwd password
```

Parameter	Description
-map <i>map</i>	Deployment map from which to export substitute configuration values.
-project <i>project</i>	Project to which the map belongs.
-varsub	Exports the substitute configuration values.

Deployer creates a file that contains the substitute configuration values. The file is named *project_map.vs* and is stored in the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\outbound directory.

If you exported substitute configuration values for scheduled tasks, open the *project_map.vs* file in an XML editor and set the task ID for each scheduled task to the task ID used on the target Integration Server.

Note:

If no substitute configuration values are specified in the deployment map, the Deployer creates a file with the complete structure but does not export any values.

Importing Substitute Configuration Variables for Integration Server Assets into a Deployment Map

Before you can import substitute configuration values into a deployment map, you must copy the exported *project_map.vs* file to the *Integration Server_directory* \instances*instance_name*\packages\WmDeployer\replicate\inbound directory on the machine that hosts the target Deployer.

You can open the *project_map.vs* file in an XML editor and edit the values before importing. For example, if you exported substitute configuration values for scheduled tasks, you must edit the file for each target Integration Server so that the task ID for each scheduled task is set to the task ID used on the target Integration Server.

Run the following command to import substituted configuration variables for Integration Server assets into a deployment map:

```
Deployer.{sh|bat} --import -varsub -vsFile project_map.vs -map map
-project project -validate {true|false}
-host host -port port -user user_name -pwd password
```

Parameter	Description
-varsub	Imports the variable substitution values.
-vsFile <i>project_map.vs</i>	File that contains the substitute configuration values to import. These files are named <i>project_map.vs</i> and are located in the following directory:

Parameter	Description
	<i>Integration Server_directory</i> <i>\instances\instance_name\packages\WmDeployer\replicate\inbound</i>
<code>-map map</code>	Deployment map into which to import the values.
<code>-project project</code>	Project that contains the map into which to import the values.
<code>-validate {true false}</code>	Whether Deployer should check the values to make sure they are valid for the target servers. If you specify <code>true</code> (validate), Deployer lists any servers that are not running on the console.

Deleting a Deployment Map from a Project

Run the following command to delete a deployment map from a project:

```
Deployer.{sh|bat} --delete -map map-project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-map map</code>	Deployment map to delete.
<code>-project project</code>	Project that contains the map to delete.

Deployment Commands

This section describes the commands to create, display information about, deploy, and delete deployment candidates and to generate checkpoints, simulate a deployment, roll back a target server, and generate reports.

Note:

You must have Deploy ACL authorization to run the commands in this section.

Creating a Deployment Candidate

Run the following command to create a deployment candidate:

```
Deployer.{sh|bat} --create -dc deployment_candidate-build build -map map
-project project-host host-port port -user user_name -pwd password
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate to create.
<code>-build build</code>	Project build to use in the deployment candidate.
<code>-map map</code>	Deployment map to use in the deployment candidate.

Parameter	Description
<code>-project project</code>	Project to which the build and map belong.

Displaying Information About a Deployment Candidate

Run the following command to display information about a deployment candidate:

```
Deployer.{sh|bat} --describe -dc deployment_candidate-project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate for which to obtain information, such as: <ul style="list-style-type: none"> ■ Name of the build and deployment map in the candidate. ■ Date the candidate was created. ■ All existing deployment reports for the candidate.
<code>-project project</code>	Project to which the deployment candidate belongs.

Deleting a Deployment Candidate

Run the following command to delete a deployment candidate:

```
Deployer.{sh|bat} --delete -dc deployment_candidate-project project
-host host -port port -user user_name -pwd password
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate to delete.
<code>-project project</code>	Project to which the deployment candidate belongs.

Generate a Checkpoint

Note:

The target servers must be available for the checkpoint generation to be successful. For more information about checkpoints, see [“Generating a Checkpoint” on page 234](#).

Run the following command to generate a checkpoint:

```
Deployer.{sh|bat} --checkpoint -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate you plan to deploy.
<code>-project project</code>	Project to which the deployment candidate belongs.
<code>-reportFilePath report_path</code>	Full path to the local directory where Deployer stores the generated checkpoint report.

Simulating a Deployment

When you run this command and simulate a deployment, Deployer generates a simulation report. Display the simulation report as instructed in [“Displaying a Simulation, Rollback, or Deployment Report” on page 265](#) and address all problems.

Note:

If you do not address all problems at this time, you will probably experience errors during deployment.

Run the following command to simulate a deployment:

```
Deployer.{sh|bat} --simulate -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate for which to simulate a deployment.
<code>-project project</code>	Project to which the deployment candidate belongs.
<code>-reportFilePath report_path</code>	Full path to the local directory where Deployer stores the generated simulation report.

Deploying

When you run this command, Deployer deploys the assets in the candidate's project build to the target servers in the candidate's deployment map. In addition, Deployer generates a deployment report. Display the deployment report as instructed in [“Displaying a Simulation, Rollback, or Deployment Report” on page 265](#).

Run the following command to deploy a deployment candidate:

```
Deployer.{sh|bat} --deploy -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -force
-reportFilePath report_path
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate to deploy.
<code>-project project</code>	Project to which the deployment candidate belongs.
<code>-force</code>	If no checkpoint exists for the deployment candidate (for example, because you chose to generate checkpoints manually, but did not do so), Deployer will not deploy unless you specify this parameter. Note: If you deploy without a checkpoint, you will not be able to roll back target servers.
<code>-reportFilePath report_path</code>	Full path to the local directory where Deployer stores the generated deployment report.

Rolling Back Target Servers

When you roll back target servers, Deployer generates a rollback report. For information about displaying the rollback report, see [“Displaying a Simulation, Rollback, or Deployment Report” on page 265](#).

Run the following command to roll back target servers:

```
Deployer.{sh|bat} --rollback -dc deployment_candidate -project project
-host host -port port -user user_name -pwd password -reportFilePath report_path
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate whose deployed assets to remove from the target servers.
<code>-project project</code>	Project to which the deployment candidate belongs.
<code>-reportFilePath report_path</code>	Full path to the local directory where Deployer stores the generated rollback report.

Listing Simulation, Rollback, and Deployment Reports

Run the following command to list simulation, rollback, and deployment reports for a deployment candidate:

```
Deployer.{sh|bat} --list -candidate deploymentReport -dc deployment_candidate
-project project -host host -port port -user user_name -pwd password
```


Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate whose simulation, deployment, and rollback reports to list.
<code>-project project</code>	Project to which the deployment candidate belongs.

Displaying a Simulation, Rollback, or Deployment Report

Run the following command to display a simulation, rollback, or deployment report for a deployment candidate:

```
Deployer.{sh|bat} --showReport -candidate deploymentReport
-dc deployment_candidate -id integerId-project project
-host host -port port -user user_name -pwd password -
```

Parameter	Description
<code>-dc deployment_candidate</code>	Deployment candidate whose simulation, deployment, or rollback report to display.
<code>id report_identifier</code>	Identifier for the report to display. Use the <code>--list</code> command (see “Listing Simulation, Rollback, and Deployment Reports” on page 264) to display report identifiers, as well as the date and time each report was generated.
<code>-project project</code>	Project to which the deployment candidate belongs.

10 Using Project Automator

■ Exporting Projects to Project Automator	268
■ Using Handles Instead of Passwords	269
■ Error Handling and Logging for Project Automator	271
■ Root Tag	271
■ Identifying Deployer	272
■ Setting Up Aliases for Source and Target Servers	273
■ Creating Projects	308
■ Running Project Automator	324

Exporting Projects to Project Automator

To configure automatic projects, you provide the necessary specifications for automated project creation in an XML file. Only the root tag and the tag that identifies Deployer are required in the XML file.

Sample XML files are provided in the *Integration Server_directory/instances/instance_name/packages/WmDeployer/config* directory. There are two files: *ProjectAutomatorSampleForRepository.xml* provides an example of a repository-based automated project, and *ProjectAutomatorSampleForRuntime.xml* shows a sample runtime-based automated project. You can also export a project you created in the GUI for use in Project Automator.

After you create a project in the GUI, you can export the project to a specification XML file that you can then use to automate your project. You specify the data to include in the specification XML file. You can export the alias, deployment set, build, map, and deployment candidate definitions associated with the project.

➤ To export a project from the Deployer user interface

1. Go to the **Tools > Export to Project Automator** page.
2. Complete the following fields:

Field	Entry
Project	Select the project to export.
Export Alias Definition	Optional. Click to export all of the alias definitions for the source and targets associated with the project.
Export Deployment and Deletion Set Definition	Optional. Click to export all of the deployment and deletion set definitions associated with the project. Exported deployment and deletion sets include the definition set and all of the associated assets.
Export Build Definition	Optional. Click to export all of the build definitions associated with the project.
Export Map Definition	Optional. Click to export all of the map definitions associated with the project. The map definition includes all target servers, target groups, and clusters that are part of the deployment map.

Field	Entry
Export Deployment Candidate Definition	Optional. Click to export all of the deployment candidate definitions associated with the project.

3. Click **Export to Project Automator**.

Deployer exports the project specification XML file to the following location:

```
Integration Server_directory
/instances/instance_name/packages/WmDeployer/replicate/outbound/projectName_ProjectAutomator.xml
```

Where *projectName* is the name of the project.

Using Handles Instead of Passwords

Project Automator uses the `pwd` attribute to store server passwords in projects. This attribute is encrypted the first time you run Project Automator. In order to avoid passing passwords in clear text the first time you run Project Automator, you can use a *password handle*. Password handles allow you to create a password on the host Integration Server along with a corresponding key (or handle) which you then store in clear text in the `pwdHandle` attribute. The handle is encrypted as an outbound password using the Password-Based Encryption (PBE) technology installed with Integration Server. For more information about how Integration Server manages outbound passwords, see *webMethods Integration Server Administrator's Guide*.

Project Automator gets the password associated to the password handle specified in the `pwdHandle` element. You create and manage password handles in the Deployer GUI. You can also delete and modify password handles as needed.

Keep the following points in mind when using password handles:

- Password handles are valid only when Project Automator is running on the same host Integration Server on which the password handles are created.
- When using password handles, Project Automator can connect only to a Deployer installed in same directory as Project Automator itself.

Creating Password Handles

Perform the following steps to create password handles.

> To create password handles

1. From the Deployer GUI running on the same server as Project Automator, click **Deployer > Password Store**.
2. Click **Create Password Store Entry**.

3. In the right-hand pane, under **Create Password Store Entry**, complete the following fields:

Field	Entry
Password Handle	<p>The name of the password handle. This is the value you will specify in the <code>pwdHandle</code> attribute in Project Automator.</p> <p>Password handles cannot contain the following illegal characters:</p> <p><code>\$ ~ / \ # & @ ^ ! % * : ; , + = > < ' ' "</code></p>
Password	The password to associate with the password handle.

4. Click **Create**.

Modifying Password Handle Associations

Perform the following steps to modify the password associated with a password handle.


> To modify the password associated with the password handle

1. Click **Deployer > Password Store**.
2. Click the password handle you want to modify from the **Password Store Entries** list.
3. In the right-hand pane, in the **New Password** field, enter the new password to associate with the password handle.
4. Click **Update**.

Deleting Password Handles

Perform the following steps to delete password handles.

> To delete password handles

1. Click **Deployer > Password Store**.
2. Click  in the **Delete** column for the password handle.
Deployer displays a confirmation dialog.
3. Click **OK** to confirm that you want to delete the password handle.

Error Handling and Logging for Project Automator

Project Automator produces a log file (ProjectAutomatorReport.xml) that is controlled by a log4j property file stored in the *Integration Server_directory* /instances/*instance_name*/packages/WmDeployer/bin directory. You can change the properties.

```
<Report>
  <Messages type="info">
    <message>message
text</message>
    <message>message
text</message>
  </Messages type="info">
  <Messages type="error">
    <message category="category" errorCode="code" deploymentSet="set name"
deploymentProject="project
name">message text</message>
  </Messages type="error">
</Report>
```

Example of an error message:

```
<message
category="projectError" errorCode="-41" deploymentSet="myDeploymentSet"
deploymentProject="TestProject">Error adding ACLs TestACL1, TestACL2
to
Deployment Set for project TestProject</message>
```

For error messages, you can write a program to parse the attribute values and take specified actions.

Root Tag

The root tag for Project Automator consists of the `<DeployerSpec>` tag and the `exitOnError` attribute, as follows:

```
<DeployerSpec exitOnError="true or false"></DeployerSpec>
```

The following table describes the attribute you can specify in the `<DeployerSpec>` tag.

Attribute	Description
<code>exitOnError</code>	Optional. Indicates how Project Automator should handle errors. Set to: <ul style="list-style-type: none"> ■ <code>true</code> to set Project Automator to report the error and terminate the first time it encounters an error.

Attribute	Description
	<ul style="list-style-type: none"> ■ <code>false</code> to set Project Automator to report errors as they occur, but continue processing. This is the default.

Identifying Deployer

You identify the Deployer on which you will perform the project tasks in the `<DeployerServer>` tag. The `<DeployerServer>` tag enables you to specify the values required to log on to the Integration Server that hosts the Deployer.

The following example shows how to use this tag:

```
<DeployerServer>
  <host>Integration
  Server host name or IP address:port</host>

  <user>user
  name</user>
  <pwd>password</pwd>
  OR <pwdHandle>handle</pwdHandle>

</DeployerServer>
```

The following table describes the attributes you can specify in the `<DeployerServer>` tag.

Attribute	Description
<code>host</code>	Host name or IP address of the server.
<code>user</code>	User name of the server.
<code>pwd</code>	Password of the server. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="704 1346 1362 1583" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
<code>pwdHandle</code>	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269.

Setting Up Aliases for Source and Target Servers

You set up aliases for both source and target servers or only target servers, target groups, and source repositories in the `<Environment>` tag. For example:

```
<Environment>
  <{webMethods
  Broker|ProcessModel|IS|MWS|Optimize|EventServer|RulesServer|EDA|APIGateway|MFT
  |DES|UniversalMessaging|TargetGroup|Repository}>
  <{broker|pm|is|mws|optimize|eventserver|rulesserver|edaserver|
  universalmessaging|rep}alias>
    tags
  </{broker|pm|is|mws|optimize|eventserver|rulesserver|edaserver|
  universalmessaging|rep}alias>
  </{webMethods Broker|ProcessModel|IS|MWS|Optimize|EventServer|RulesServer|EDA|
  UniversalMessaging|TargetGroup|Repository}>
</Environment>
```

The following sections describe each tag within the `<Environment>` tag in detail.

If Deployer already contains an alias with the same name as one you define, Deployer overwrites the alias.

Note:

The credentials for `<user>`, `<pwd>`, and `<pwdHandle>` asset tags must be those for a user with Administrator ACL authorization or for a user that belongs to a group that has Internal, Developer, and Deployer Admin ACLs to create Deployer runtime aliases and projects.

Setting Up Aliases for Source Repositories

For repository-based deployment, you define the repository as the source server. This location identifies the repository directory from which the assets should be deployed.

Note:

You can set up aliases for source repositories for repository-based deployment only.

```
<Repository>
  <repalias name="name">
    <type>FlatFile</type>
    <urlOrDirectory>directory_location</urlOrDirectory>
    <createIndex>true or false</createIndex>
    <Test>true or false</Test>
  </repalias>
</Repository>
```

For more information about the values to supply for the following attributes, see [“Connecting to a Source Repository” on page 69](#).

Attribute	Description
repalias name	The name to use for the repository alias. This attribute corresponds to the Name field.
type	The type of repository file. Set to <code>FlatFile</code> .
urlOrDirectory	The full path of the repository directory in which the composites are located. This attribute corresponds to the File Directory field.
createIndex	<p>Specifies whether Deployer should create the index.</p> <p>Set to:</p> <ul style="list-style-type: none"> ■ <code>true</code> to create the index in the source repository. ■ <code>false</code> if you do not want to create the index in the source repository. <p>Default is <code>false</code>.</p>
test	<p>Specifies whether Deployer should test the connection to the source repository. Set to:</p> <ul style="list-style-type: none"> ■ <code>true</code> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <code>false</code> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target ActiveTransfer Server Instances

The following example illustrates how to set up aliases for target ActiveTransfer Server instances.

```
<MFT>
  <mftalias name="APP_Target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
```

```

    <pwd>password</pwd>
OR <pwdHandle>handle</pwdHandle>

    <useSSL>true/false</useSSL>

    <version>version_number</version>

    <Test>true/false</Test>

</mftalias>
</MFT>

```

For information on the values to supply for the following tags, see [“Connecting to a Target ActiveTransfer Server” on page 70](#).

Attribute	Description
mftalias	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password for the user account. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="812 1270 1461 1501" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
useSSL	Specifies whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> ■ <code>true</code> to use SSL to connect to the server.

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	Version of the server. This attribute corresponds to the Version field.
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Target AgileApps Cloud Deployment Endpoints

The following example illustrates how to set up aliases for target AgileApps Cloud deployment endpoints.

```
<AgileApps>
  <agileappsalias name="test">
    <agileAppsServerURL>asdasd</agileAppsServerURL>
    <user>asdasd</user>
    <pwd>asdasd</pwd>
    <!--<pwdHandle>$(PasswordHandle)</pwdHandle-->
    <version>10.7</version>
  <Test>false</Test>
</agileappsalias>
</AgileApps>
```

For information on the values to supply for the following tags, see [“Connecting to a Target AgileApps Cloud Server” on page 82](#).

Attribute	Description
agileappsalias name	Name to assign to the server. This attribute corresponds to the Name field.
agileAppsServerUrl	The URL of the AgileApps Cloud server. This attribute corresponds to the Server URL field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User Name field.
pwd	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <div data-bbox="899 737 1461 976" style="background-color: #f0f0f0; padding: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
version	Version of the server. This attribute corresponds to the Version field.
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Target API Gateway Servers

The following example illustrates how to set up aliases for target API Gateway servers.

```

<APIGateway>
  <apigatewayalias name="APP_Target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd>
    OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </apigatewayalias>
</APIGateway>

```

For information on the values to supply for the following tags, see [“Connecting to a Target API Gateway Server” on page 71](#).

Attribute	Description
apigatewayalias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="721 1556 1362 1793" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating

Attribute	Description
	a password handle, see “Using Handles Instead of Passwords” on page 269.
useSSL	<p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	Version of the server. This attribute corresponds to the Version field.
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Target Application Platform Deployment Endpoints

The following example illustrates how to set up aliases for target Application Platform deployment endpoints.

```

<ApplicationPlatform>
  <applicationplatformalias name="APP_Target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd>
    OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
  </applicationplatformalias>
</ApplicationPlatform>

```

```
<Test>true/false</Test>
</applicationplatformalias>
</ApplicationPlatform>
```

For information on the values to supply for the following tags, see [“Connecting to a Target Application Platform Server” on page 75](#).

Attribute	Description
application platformalias name	Name to assign to the Application Platform deployment endpoint. This attribute corresponds to the Name field.
host	Host name or IP address of the Software AG Platform Manager runtime. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <div data-bbox="716 1163 1365 1402" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
useSSL	<p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>

Attribute	Description
version	Version of the server. This attribute corresponds to the Version field.
Test	Specifies whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Target Digital Event Services Servers

The following example illustrates how to set up aliases for target Digital Event Services (DES) deployment endpoints.

```
<DES>
  <desalias name="DES_Target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd>
    OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </desalias>
</DES>
```

For information on the values to supply for the following tags, see [“Connecting to a Deployment Endpoint for Digital Event Services” on page 72](#).

Attribute	Description
desalias	Name to assign to the server. This attribute corresponds to the Name field.

Attribute	Description
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="716 737 1377 968" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
useSSL	Specifies whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> <li data-bbox="716 1251 1268 1276">■ <i>true</i> to use SSL to connect to the server. <li data-bbox="716 1308 1377 1375">■ <i>false</i> to connect to the server without any client authentication. This attribute corresponds to the Use SSL field.
version	Version of the server. This attribute corresponds to the Version field.
Test	Specifies whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> <li data-bbox="716 1654 1377 1856">■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271.

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>false</i> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Source and Target webMethods Brokers

You can set up aliases for source and target Broker Servers for either basic authentication, SSL authentication, or neither.

Basic Authentication

The following example illustrates how to set up a Broker alias that uses basic authentication.

```
<Broker>
  <brokeralias name="alias name">
    <brokerName>Broker name</brokerName>
    <clientGroup>client group</clientGroup>
    <host>Broker server host</host>
    <port>Broker Server port</port>
    <useBasicAuth>true</useBasicAuth>
      <user>basic authorization user name</user>
      <pwd>basic authorization password</pwd> OR <pwdHandle>handle</pwdHandle>

    <version>version_number</version>
    <context>JNDI context</context>
    <Test>true/false</Test>
  </brokeralias>
</Broker>
```

For detailed information on the values to supply for the following attributes, see [“Connecting to webMethods Broker” on page 28](#).

Attribute	Description
brokeralias name	Name to assign to the Broker Server. This attribute corresponds to the Name field.
brokerName	Name of the source or target webMethods Broker. This attribute corresponds to the BrokerName field.
clientGroup	Client group Deployer should use to access the source or target Broker Server. For target Broker Servers, specify <i>admin</i> . This attribute corresponds with the Client Group field.
host	Host name or IP address of the Broker Server. This attribute corresponds to the Host field.

Attribute	Description
port	Port for the Broker Server. This attribute corresponds to the Port field.
useBasicAuth	Set to <i>true</i> to connect to the Broker Server using basic authentication. This attribute corresponds to the Client Authentication > Username/Password check box.
user	Basic authentication user name. This attribute corresponds to the Username field.
pwd	<p>Basic authentication password. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <div data-bbox="737 737 1365 974" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
version	Version of the Broker Server. This attribute corresponds to the Version field.
context	JNDI context. Required if the Broker Server serves as a JNDI provider. This attribute corresponds to the Context field.
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

SSL Authentication

The following example illustrates how to set up a Broker alias that uses SSL authentication.

```
<Broker>
  <brokeralias name="alias name">
    <brokerName>Broker name</brokerName>
    <clientGroup>client group</clientGroup>
    <host>Broker server host</host>
    <port>Broker Server port</port>
    <useSSL>true</useSSL>
    <version>version_number</version>
    <keyStoreType>Deployer keystore type</keyStoreType>

    <keyStorePath>Deployer keystore path</keyStorePath>

    <keyStorepassword>Deployer keystore password</keyStorepassword>
    <trustStoreType>Deployer trust store type</trustStoreType>
    <trustStorePath>Deployer truststore path</trustStorePath>
    <context>JNDI context</context>
    <Test>true/false</Test>
  </brokeralias>
</Broker>
```

For detailed information on the values to supply for the following attributes, see [“Connecting to webMethods Broker” on page 28](#).

Attribute	Description
brokeralias name	Name to assign to the Broker Server. This attribute corresponds to the Name field.
brokerName	Name of the source or target webMethods Broker. This attribute corresponds to the BrokerName field.
clientGroup	Client group Deployer should use to access the source or target Broker Server. For target Broker Servers, specify <i>admin</i> . This attribute corresponds with the Client Group field.
host	Host name or IP address of the Broker Server. This attribute corresponds to the Host field.
port	Port for the Broker Server. This attribute corresponds to the Port field.
useSSL	Set to <i>true</i> to connect to the Broker Server using SSL authentication. This attribute corresponds to the Client Authentication > SSL check box.

Attribute	Description
version	Version of the Broker Server. This attribute corresponds to the Version field.
keyStoreType	File type of Deployer's keystore file. This attribute corresponds to the Keystore Type field.
keyStorePath	Full path to Deployer's keystore file. This attribute corresponds to the DeployerKeystore field.
keyStore password	Password that Deployer uses to access its keystore file. This attribute corresponds to the Keystore Password field.
trustStoreType	File type of Deployer's truststore file. This attribute corresponds to the Truststore Type field.
trustStorePath	Full path to Deployer's truststore file. This attribute corresponds to the DeployerTruststore field.
context	JNDI context. Required if the Broker Server serves as a JNDI provider. This attribute corresponds to the Context field.
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

No Authentication

The following example illustrates how to set up a Broker alias that does not use client authentication.

```
<Broker>
  <brokeralias name="alias name">
    <brokerName>Broker name</brokerName>
    <clientGroup>client group</clientGroup>
    <host>Broker server host</host>
```

```

<port>Broker Server port</port>
<useSSL>>false</useSSL>
<version>version number</version>
<context>JNDI context</context>
<Test>>true/false</Test>
</brokeralias>
</Broker>

```

For detailed information about the values to supply for the following attributes, see [“Connecting to webMethods Broker”](#) on page 28.

Attribute	Description
brokeralias name	Name to assign to the Broker Server. This attribute corresponds to the Name field.
brokerName	Name of the source or target webMethods Broker. This attribute corresponds to the BrokerName field.
clientGroup	Client group Deployer should use to access the source or target Broker Server. For target Broker Servers, type <i>admin</i> . This attribute corresponds to the Client Group field.
host	Host name or IP address of the Broker Server. This attribute corresponds to the Host field.
port	Port for the Broker Server. This attribute corresponds to the Port field.
useSSL	Set to <i>false</i> to connect to the Broker Server without any client authentication. This attribute corresponds to the Client Authentication > None check box.
version	Version of the Broker Server. This attribute corresponds to the Version field.
context	JNDI context. Required if the Broker Server serves as a JNDI provider. This attribute corresponds to the Context field.
Test	Specifies whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute

Attribute	Description
	<p>of the <DeployerSpec> tag. For more information, see “Root Tag” on page 271.</p> <ul style="list-style-type: none"> ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Source and Target Process Model Servers

The following example illustrates how to set up aliases for source and target process model servers.

```
<ProcessModel>
  <palias name="alias name">
    <host>ProcessModel Server host</host>
    <port>ProcessModel Server port</port>
    <user>user name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>

    <useSSL>true/false</useSSL>
    <version>version_number</version>

    <Test>true/false</Test>
  </palias>
</ProcessModel>
```

For detailed information about the values to supply for the following attributes, see [“Connecting to BPM Process Model Servers” on page 26](#).

Attribute	Description
pmalias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either pwd or pwdHandle.

Attribute	Description
	<p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p>
pwdHandle	<p>The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code>. For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269.</p>
useSSL	<p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	<p>Version of the server. This attribute corresponds to the Version field.</p>
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Source and Target Integration Servers

The following example illustrates how to set up aliases for source and target Integration Servers.

```
<IS>
  <isalias name="alias name">
    <host>Integration Server host</host>
    <port>Integration Server port</port>
    <user>user name</user>
    <pwd>password</pwd>
  OR <pwdHandle>handle</pwdHandle>
```

```

<useSSL>true/false</useSSL>
<version>version_number</version>
<installDeployerResource>true/false</installDeployerResource>
<Test>true/false</Test>
<executeACL>acl</executeACL>
</isalias>
</IS>

```

For information on the values to supply for the following tags, see [“Connecting to Integration Server and Trading Networks”](#) on page 23.

Attribute	Description
isalias name	Name to assign to the server.
host	Host name or IP address of the server.
port	Port for the server.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server.
pwd	Password associated with the user name. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="776 972 1362 1213" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269.
useSSL	Specifies whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication.
version	Version of the server.
install Deployer Resource	Specifies whether Deployer should install the <code>WmDeployerResource</code> package on each Integration Server server that will run the process steps. Set to:

Attribute	Description
Test	<ul style="list-style-type: none"> <li data-bbox="873 258 1260 289">■ <i>true</i> to install the package. <li data-bbox="873 317 1385 348">■ <i>false</i> to avoid installing the package. <p data-bbox="873 373 1429 443">Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> <li data-bbox="873 470 1477 709">■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. <li data-bbox="873 737 1477 804">■ <i>false</i> to create the source alias without testing the connection to the target server.
executeACL	<p data-bbox="873 835 1477 930">Optional. Specifies the ACL for the alias. If no value is specified, Deployer assigns the alias to the Administrators ACL by default.</p>

Setting Up Aliases for Source and Target My webMethods Servers

The following example illustrates how to set up aliases for source and target My webMethods Servers.

```

<MWS>
  <mwsalias name="alias name">
    <host>My webMethods Server host</host>
    <port>My webMethods Server port</port>
    <user>user name</user>
    <pwd>password</pwd>
  OR <pwdHandle>handle</pwdHandle>
    <excludeCoreTaskEngineDependencies>true/false
    </excludeCoreTaskEngineDependencies>
    <cacheTimeOut>time</cacheTimeOut>
    <includeSecurityDependencies>true/false</includeSecurityDependencies>
    <rootFolderAliases>folder, folder, folder...</rootFolderAliases>
    <maximumFolderObjectCount>count</maximumFolderObjectCount>
    <enableAdditionalLogging>true/false</enableAdditionalLogging>
    <maxFolderDepth>number_of_assets</maxFolderDepth>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </mwsalias>
</MWS>

```

For information on the values to supply for the following tags, see [“Connecting to My webMethods Server” on page 25](#).

Attribute	Description
mwsalias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <div data-bbox="886 1062 1362 1371" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
excludeCore TaskEngine Dependencies	<p>Specifies whether to exclude Task Engine portlets from the dependencies list for task application assets. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to exclude the portlets. ■ <i>false</i> to include the portlets.

Attribute	Description
	This attribute corresponds to the Exclude Core Task Engine Dependencies field.
cacheTimeout	Length of time queries should remain in the cache unless the cache capacity is exceeded. This attribute corresponds to the Cache Timeout field.
include Security Dependencies	<p>Specifies whether to include the following in the dependencies list for My webMethods Server assets when creating an MWS deployment set:</p> <ul style="list-style-type: none"> ■ Security realms that contain the assets. ■ User/group/role references in the assets' security ACLs. <p>Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to include the assets. ■ <i>false</i> to exclude the assets. <p>This attribute corresponds to the Include security dependencies field.</p>
rootFolder Aliases	My webMethods Server aliases to use as root folders when selecting pages to deploy. Separate the folders using commas. This attribute corresponds to the Root folder aliases field.
maximumFolder ObjectCount	Maximum number of assets to display within My webMethods Server folders when you are defining and choosing assets to include in an MWS deployment set. This attribute corresponds to the Maximum Folder Object Count field.
enable Addtional Logging	Specifies whether to log debug information about selected assets to source My webMethods Server logs, and assets that Deployer deploys to target My webMethods Server logs. Set to:

Attribute	Description
maxFolderDepth	<ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Enable additional MWS logging field.</p>
useSSL	<p>Maximum number of assets to display within My webMethods Server folders when you are defining and choosing assets to include in an MWS deployment set. This attribute corresponds to the Maximum Folder Depth field.</p> <p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	<p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <p>Version of the server. This attribute corresponds to the Version field.</p>
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Source and Target Optimize Servers

The following example illustrates how to set up aliases for source and target Optimize servers.

```
<Optimize>
  <optimizealias name="alias name">
    <host>Integration Server host</host>
    <port>Integration Server port</port>
    <user>user name</user>
    <pwd>password</pwd> OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </optimizealias>
</Optimize>
```

For information on the values to supply for the following tags, see [“Connecting to Optimize Servers” on page 30](#).

Attribute	Description
optimizealias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="841 1455 1458 1696" style="background-color: #f0f0f0; padding: 5px; margin-top: 5px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .

Attribute	Description
useSSL	<p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	<p>Version of the server. This attribute corresponds to the Version field.</p>
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Event Servers

The following example illustrates how to set up aliases for target Event Servers.

Note:

Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.

```

<EventServer>
  <eventserveralias name="event_server_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd>
  OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </eventserveralias>
</EventServer>

```



```
</eventserveralias>
</EventServer>
```

For information on the values to supply for the following tags, see [“Connecting to a Target Event Server” on page 77](#).

Attribute	Description
eventserver alias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="857 1039 1461 1276" style="background-color: #f0f0f0; padding: 5px; margin-top: 5px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
useSSL	Specifies whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. This attribute corresponds to the Use SSL field.
version	Version of the server. This attribute corresponds to the Version field.

Attribute	Description
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target EDA Deployment Endpoints

The following example illustrates how to set up aliases for target EDA deployment endpoints.

```

<EDA>
  <edaserveralias name="EDA_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd>
    OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </edaserveralias>
</EDA>

```

For information on the values to supply for the following tags, see [“Connecting to a Deployment Endpoint for EDA” on page 85](#).

Attribute	Description
<code>edaserveralias name</code>	Name to assign to the EDA deployment endpoint. This attribute corresponds to the Name field.
<code>host</code>	Host name or IP address of the Software AG Platform Manager runtime. This attribute corresponds to the Host field.

Attribute	Description
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <div data-bbox="760 604 1461 810" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
useSSL	<p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	Version of the server. This attribute corresponds to the Version field.
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the target alias without testing the connection to the target server.

Setting Up Aliases for Target Business Rules Integration Servers

The following example illustrates how to set up aliases for source and target Business Rules Integration Servers.

```
<RulesServer>
  <rulesserveralias name="rules_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd>
    OR <pwdHandle>handle</pwdHandle>
    <useSSL>true/false</useSSL>
    <version>version_number</version>
    <Test>true/false</Test>
  </rulesserveralias>
</RulesServer>
```

For information on the values to supply for the following tags, see [“Connecting to a Target Business Rules Integration Server” on page 85](#).

Attribute	Description
rulesserver alias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> .

Note:
Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the

Attribute	Description
	passwords in the XML file and run Project Automator to encrypt the passwords again.
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
useSSL	Specifies whether Deployer should use SSL to connect to the server. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. This attribute corresponds to the Use SSL field.
version	Version of the server. This attribute corresponds to the Version field.
Test	Specifies whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Business Rules My webMethods Servers

The following example illustrates how to set up aliases for source and target Business Rules My webMethods Servers.

```
<RulesServerMWS>
  <rulesmwsserveralias name="rules_mws_target">
    <host>host_name</host>
    <port>port_number</port>
    <user>user_name</user>
    <pwd>password</pwd>
  OR <pwdHandle>$(PasswordHandle)</pwdHandle>
```

```

<useSSL>true/false</useSSL>

<version>version_number</version>
<rootContext>mws_root_context</rootContext>

<Test>true/false</Test>

</rulesmwsserveralias>
</RulesServerMWS>

```

For information on the values to supply for the following tags, see [“Connecting to a Target Business Rules My webMethods Server”](#) on page 86.

Attribute	Description
rulesmwsserveralias name	Name to assign to the server. This attribute corresponds to the Name field.
host	Host name or IP address of the server. This attribute corresponds to the Host field.
port	Port for the server. This attribute corresponds to the Port field.
user	Optional. User name for a user account with Administrator authority that Deployer can use to access the server. This attribute corresponds to the User field.
pwd	<p>Password associated with the user name. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code>.</p> <div data-bbox="597 1142 1360 1331" style="background-color: #f0f0f0; padding: 5px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269.
useSSL	<p>Specifies whether Deployer should use SSL to connect to the server. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to use SSL to connect to the server. ■ <i>false</i> to connect to the without any client authentication. <p>This attribute corresponds to the Use SSL field.</p>
version	Version of the server. This attribute corresponds to the Version field.

Attribute	Description
rootContext	The root context path for the server (http://:host_name:port_number/root_context). For example, http://:localhost:8585/mws.
Test	Specifies whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Universal Messaging Servers

The following example illustrates how to set up aliases for target Universal Messaging servers for basic authentication, SSL authentication, or neither.

Basic Authentication

The following example illustrates how to set up a Universal Messaging server alias that uses basic authentication.

```
<UniversalMessaging>
  <universalmessagingalias name="server_target">
    <realmURL>URL</realmURL>
    <useBasicAuth>true</useBasicAuth>
    <version>version</version>
    <user>basic authorization user name</user>
    <pwd>basic authorization password</pwd> OR <pwdHandle>handle</pwdHandle>
    <Test>true/false</Test>
  </universalmessagingalias>
</UniversalMessaging>
```

For detailed information on the values to supply for the following attributes, see [“Connecting to a Target Universal Messaging Server” on page 83](#).

Attribute	Description
universalmessagingalias name	Name to assign to the server. This attribute corresponds to the Name field.
realmURL	The URL of the Universal Messaging realm server. This attribute corresponds to the Realm URL field.

Attribute	Description
useBasicAuth	Set to <i>true</i> to connect to the Universal Messaging server using basic authentication. This attribute corresponds to the Client Authentication > Basic Authentication option.
version	Version of the server. This attribute corresponds to the Version field.
user	Basic authentication user name. This attribute corresponds to the Username field.
pwd	Basic authentication password. This attribute corresponds to the Password field. You must specify either <code>pwd</code> or <code>pwdHandle</code> . <div data-bbox="755 737 1365 976" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Project Automator encrypts passwords the first time it runs. If you must change the passwords in the future, change the passwords in the XML file and run Project Automator to encrypt the passwords again.</p> </div>
pwdHandle	The password handle. You must specify either <code>pwd</code> or <code>pwdHandle</code> . For more information about creating a password handle, see “Using Handles Instead of Passwords” on page 269 .
Test	Specifies whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

SSL Authentication

The following example illustrates how to set up a Universal Messaging server alias that uses SSL authentication.

```
<UniversalMessaging>
  <universalmessagingalias name="server_target">
    <realmURL>URL</realmURL>
    <useSSL>true</useSSL>
  </universalmessagingalias>
</UniversalMessaging>
```



```

<version>version</version>
<keyStorePath>Deployer keystore path</keyStorePath>
<keyStorepassword>Deployer keystore password</keyStorepassword>
<trustStorePath>Deployer truststore path</trustStorePath>
<trustStorepassword>Deployer truststore password</trustStorepassword>
<Test>true/false</Test>
</universalmessagingalias>
</UniversalMessaging>

```

For detailed information on the values to supply for the following attributes, see [“Connecting to a Target Universal Messaging Server” on page 83](#).

Attribute	Description
universalmessagingalias name	Name to assign to the server. This attribute corresponds to the Name field.
realmURL	The URL of the Universal Messaging realm server. This attribute corresponds to the Realm URL field.
useSSL	Set to <i>true</i> to connect to the Universal Messaging server using SSL authentication. This attribute corresponds to the Client Authentication > SSL check box.
version	Version of the Universal Messaging server. This attribute corresponds to the Version field.
keyStorePath	Full path to Deployer's keystore file. This attribute corresponds to the DeployerKeystore field.
keyStore password	Password that Deployer uses to access its keystore file. This attribute corresponds to the Keystore Password field.
trustStorePath	Full path to Deployer's truststore file. This attribute corresponds to the DeployerTruststore field.
trustStorepassword	Password that Deployer uses to access its truststore file. This attribute corresponds to the Truststore Password field.
Test	Specifies whether Deployer should test the connection to the servers. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271.

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>false</i> to create the source alias without testing the connection to the target server.

No Authentication

The following example illustrates how to set up a Universal Messaging server alias that does not use client authentication.

```
<UniversalMessaging>
  <universalmessagingalias name="server_target">
    <realmURL>URL</realmURL>
    <version>version</version>
    <Test>true/false</Test>
  </universalmessagingalias>
</UniversalMessaging>
```

For detailed information on the values to supply for the following attributes, see [“Connecting to a Target Universal Messaging Server” on page 83](#).

Attribute	Description
universalmessagingalias name	Name to assign to the server. This attribute corresponds to the Name field.
realmURL	The URL of the Universal Messaging realm server. This attribute corresponds to the Realm URL field.
version	Version of the Universal Messaging server. This attribute corresponds to the Version field.
Test	<p>Specifies whether Deployer should test the connection to the servers. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to test the connection to the target server. If Project Automator cannot ping the target server, it registers an error and handles the error according to the <code>exitOnError</code> attribute of the <code><DeployerSpec></code> tag. For more information, see “Root Tag” on page 271. ■ <i>false</i> to create the source alias without testing the connection to the target server.

Setting Up Aliases for Target Groups

Use the `<TargetGroup>` tag to define the aliases to include in target groups as follows:

```
<TargetGroup
description="target
```

```

group description" isLogicalCluster="true or
false" name="alias name" type="runtime_type" version="version_number">
<alias>server
alias</alias>
  <alias>server
alias</alias>
  <cluster name="cluster name">server,server,server...</cluster>

  <cluster name="cluster name">server,server,server...</cluster>

</TargetGroup>

```

For information on the values to supply for the following attributes, see [“Creating Target Groups” on page 219](#).

Attribute	Description
description	Description of the target group. This attribute corresponds to the Description field.
isLogical Cluster	<p>(Runtime-based deployment only.) Specifies whether Deployer should automatically roll back the deployment on all servers in the group if deployment fails on any of the servers in the target group. This attribute corresponds to the Rollback All on Failure field. Set to:</p> <ul style="list-style-type: none"> ■ <i>true</i> to roll back all of the servers in the target group if deployment fails on one of the servers in the target group. ■ <i>false</i> to keep Deployer from rolling back all of the servers in the target group if deployment to one of the servers encounters a failure. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: The <code>isLogicalCluster</code> attribute is valid for runtime-based deployment only. Deployer ignores this attribute for repository-based deployment.</p> </div>
name	Name to use for the target group. This attribute corresponds to the Name field.
type	<p>The runtime type of the target group. Set to:</p> <ul style="list-style-type: none"> ■ <i>MFT</i> ■ <i>APIGateway</i> ■ <i>Broker</i> ■ <i>DES</i> ■ <i>IS</i> ■ <i>MWS</i>

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>Optimize</i> ■ <i>ProcessModel</i> ■ <i>RULES</i> ■ <i>EventServer</i> <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> </div> <ul style="list-style-type: none"> ■ <i>EDA</i> ■ <i>UniversalMessaging</i>
version	Version of the server. This attribute corresponds to the Version field.
alias	The aliases of the servers to include in the target group. Each alias should have its own alias attribute.
cluster name	A list of Integration Server or process model clusters.

Creating Projects

You create projects in the <Projects> tag as follows:

```
<Projects
projectPrefix="string">
  <Project overwrite="true" name="project_name"
description=project_description">
    <ProjectProperties>
      <Property name= "projectLocking">true/false</Property>
      <Property name= "concurrentDeployment">true/false</Property>
      <Property name= "ignoreMissingDependencies">true/false</Property>
      <Property name= "isTransactionalDeployment">true/false</Property>
    </ProjectProperties>
    <{DeploymentSet|DeletionSet}>tags</{DeploymentSet|DeletionSet}>
    <{Component|Composite}>tags</{Component|Composite}>
  </Project>
</Projects>
```

Note:

You can specify attributes in the <DeletionSet> tag for repository-based deployment only.

Note:

The <Component> and <Composite> tags define the components and composites for repository-based deployment projects only. They are not used for runtime-based deployment projects.

The <Projects> tag can contain the following attribute:

Attribute	Description
projectPrefix	Optional. Specify a prefix for projects created using Project Automator. For example, if you specified <Projects projectPrefix="Auto_">, the names of projects created using Project Automator would be prefixed by "Auto_".

The <Projects> tag can contain several <Project> tags, one for each project you want Project Automator to create. You can specify the following attributes for the <Project> tag:

Attribute	Description
overwrite	Determines how Project Automator proceeds if Deployer already contains a project with the same name as one you define. Set to: <ul style="list-style-type: none"> ■ <i>true</i> to overwrite the project. ■ <i>false</i> keep the project, write an error, and continue to the next <Project> tag.
name	Name of the project.
description	Description of the project.

For repository-based projects, the <ProjectProperties> tag contains the individual project properties that you can edit for the project. If no project properties are specified, the project adopts the settings specified for all projects. For more information about setting project settings for all projects, see ["Settings for All Projects" on page 202](#).

Attribute	Description
projectLocking	Indicates whether locking is enabled or disabled for the project. When set to <i>true</i> , locking is enabled for the project.
concurrent Deployment	Indicates whether Deployer deploys assets concurrently. When set to <i>true</i> concurrent deployment is enabled.
ignoreMissing Dependencies	Indicates whether Deployer ignores missing dependencies. When set to <i>true</i> , Deployer ignores missing dependencies for the project. If this attribute is set to <i>false</i> and the project contains missing dependencies, deployment fails.

Attribute	Description
isTransactional Deployment	Indicates whether Deployer automatically creates a checkpoint prior to delivering and activating deployment and deletion sets. When set to <i>true</i> , transactional deployment is enabled. When transactional deployment is enabled and activation fails, Deployer triggers a roll back automatically and restores the target servers to the state of the prior activation.

The following sections describe the tasks you can perform using tags of the parent <Project> tag.

Defining Deployment and Deletion Sets for Runtime-Based Deployment

You define the deployment and deletion sets for a runtime-based project using the <DeploymentSet> tag. Each tag can include these attributes:

```
<DeploymentSet
name="set name" description="set description" type="runtime_type"
mode="{Deploy|Delete}" srcAlias="{source server} alias"
defaultDependencyAction="{Add|Fulladd|Exists|Ignore}"></DeploymentSet>
```

The following table describes the attributes in the <DeploymentSet> tag:

Attribute	Description
name	Name of the deployment or deletion set. For more information, see the description of the Name field in “Creating a Deployment Set from Runtime Assets” on page 32 or “Creating a Deletion Set” on page 47 .
description	Optional. Description for the deployment or deletion set. For more information, see the description of the Description field in “Creating a Deployment Set from Runtime Assets” on page 32 or “Creating a Deletion Set” on page 47 .
type	Runtime type of the server that contains the assets to deploy or delete. Set to: <ul style="list-style-type: none"> ■ <i>Broker</i> ■ <i>IS</i> (For Integration Server and Trading Networks servers.) ■ <i>MWS</i> (Deployment sets only.)

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>Optimize</i> (Deployment sets only.) ■ <i>ProcessModel</i> (Deployment sets only.) ■ <i>RULES</i> (Deployment sets only.) ■ <i>EventServer</i> (Deployment sets only.) <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> </div> <ul style="list-style-type: none"> ■ <i>EDA</i> (Deployment sets only.) <p>For more information, see the description of the Type field in “Creating a Deployment Set from Runtime Assets” on page 32 or “Creating a Deletion Set” on page 47.</p>
packageRegExp	<p>(Optional.) Text that the package names must contain in order to be listed. For more information, see the description of the Packages field in “Creating a Deployment Set from Runtime Assets” on page 32 or “Creating a Deletion Set” on page 47.</p>
otherRegExp	<p>(Optional.) The number of assets to display in the list. See the All other assets field in “Creating a Deployment Set from Runtime Assets” on page 32 or “Creating a Deletion Set” on page 47.</p>
mode	<p>Specifies whether Deployer should create a deployment or deletion set for the project. Set to:</p> <ul style="list-style-type: none"> ■ <i>deploy</i> to define a deployment set. ■ <i>delete</i> to define a deletion set. <p>For example, to define a deployment set, specify <i>deploy</i> for mode as follows:</p>
	<pre style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <DeploymentSet name="depSet1" description="depAssets" packageRegExp="" otherRegExp="" type="IS" mode="deploy or delete" srcAlias="ISserver31,ISserver41" defaultDependencyAction="Add" tnTreeNodeCount="1000"></DeploymentSet></pre>
srcAlias	<p>The server alias for the source server (for deployment sets) or target server (for deletion sets). This attribute corresponds with the Name field described for the</p>

Attribute	Description
default Dependency Action	<p data-bbox="618 260 1294 327">runtime server type in “Connecting to webMethods Source and Target Runtimes” on page 22.</p> <p data-bbox="618 354 1294 422">(Optional.) Specifies how Deployer should handle unresolved dependencies. Set to:</p> <ul data-bbox="618 449 1294 701" style="list-style-type: none"> <li data-bbox="618 449 1068 474">■ <i>add</i> to add the referenced asset. <li data-bbox="618 506 1294 606">■ <i>fulladd</i> to add the entire package that contains the referenced asset. This value is valid for Integration Server assets only. <li data-bbox="618 638 1294 701">■ <i>exists</i> to specify that the referenced asset exists on the target server. <div data-bbox="670 716 1276 852" style="background-color: #f0f0f0; padding: 5px;"> <p data-bbox="670 730 748 756">Note:</p> <p data-bbox="670 766 1230 833">If the dependent asset does not exist on the target server, deployment will fail.</p> </div> <ul data-bbox="618 869 1208 900" style="list-style-type: none"> <li data-bbox="618 869 1208 900">■ <i>ignore</i> to ignore unresolved dependencies.

The tags for each type of asset vary. See the sample XML file `ProjectAutomatorSample.xml` in the `Integration Server_directory/instances/instance_name/packages/WmDeployer/config` directory for examples for each type of asset. The following additional notes are provided.

Assets	Notes
All	If an asset is located in a folder, the asset tag must include the <code>folder</code> attribute.
Integration Server	<ul data-bbox="553 1245 1205 1850" style="list-style-type: none"> <li data-bbox="553 1245 1205 1312">■ On <code><Port></code> asset tags, set the <code>protocol</code> attribute to one of the following: <ul data-bbox="602 1339 781 1665" style="list-style-type: none"> <li data-bbox="602 1339 727 1365">■ <i>HTTP</i> <li data-bbox="602 1396 704 1421">■ <i>FTP</i> <li data-bbox="602 1453 743 1478">■ <i>HTTPS</i> <li data-bbox="602 1509 721 1535">■ <i>FTPS</i> <li data-bbox="602 1566 721 1591">■ <i>Email</i> <li data-bbox="602 1623 781 1648">■ <i>FilePolling</i> <li data-bbox="553 1692 1105 1755">■ Package asset tags require the attribute <code>package="name"</code>. <li data-bbox="553 1787 1149 1850">■ Scheduled task assets require the attribute <code>filterBy="name"</code>.

Assets	Notes
Trading Networks	Sets that include Trading Networks assets must also include the <code>tnTreeNodeCount="number"</code> attribute. This attribute specifies the number of Trading Networks assets Deployer should display. For more information, see the description of the Maximum TN Assets to Display field in “Creating a Deployment Set from Runtime Assets” on page 32 or “Creating a Deletion Set” on page 47.
My webMethods Server	<p><Rule> asset tags require the attribute <code>folder</code> set to:</p> <ul style="list-style-type: none"> ■ <i>Shell Rules</i> ■ <i>Skin Rules</i> ■ <i>Start Page Rules</i> ■ <i>Rendering Rules</i> ■ <i>Login Page Rules</i> ■ <i>Task Rules\Global Task Rules\Schedule Rules</i> ■ <i>Task Rules\Global Task Rules\Trigger Rules</i>

Defining a Deployment Set for Repository-Based Deployment

You define deployment sets for repository-based deployment as follows:

```
<DeploymentSet
autoResolve="full|partial|ignore" description="description"
name="deployment_set" srcAlias="repository">
  <Composite name="name"
srcAlias="repository_alias"
type="type"/>
  <Component
componentType="component_type" compositeName="name" name="name"
srcAlias="repository_alias" type="runtime_type"/>
```

The `<DeploymentSet>` tag must include the `autoResolve`, `description`, `name`, and `srcAlias` attributes. The following table describes each of these attributes.

Attribute	Description
<code>autoResolve</code>	<p>Specifies how Deployer should resolve unresolved dependencies in the deployment set. Set to:</p> <ul style="list-style-type: none"> ■ <i>full</i> to resolve the unresolved dependencies for the deployment set by adding the entire referenced composite. This setting corresponds to the Auto resolve

Attribute	Description
	<p>by Composite field in the GUI. For more information, see “Resolving Dependencies Automatically” on page 91.</p> <ul style="list-style-type: none"> ■ <i>partial</i> to resolve the unresolved dependencies for the deployment set by adding only the referenced assets. Adding only the referenced assets, instead of the entire referenced composite, is referred to as <i>partial deployment</i>. You can use partial deployment for only Integration Server, CloudStreams, Trading Networks, or Broker (including JNDI) assets. This setting corresponds to the Auto resolve by Asset field in the GUI. For more information, see “Resolving Dependencies Automatically” on page 91. ■ <i>ignore</i> to ignore unresolved dependencies. This corresponds to manually resolving dependencies by setting the Unset/Add/Ignore column to Ignore for an asset. For more information, see “Resolving Dependencies Manually” on page 92.
description	This attribute corresponds with the Description field described in “Creating a Deployment Set from Runtime Assets” on page 32 .
name	Name of the deployment set. This attribute corresponds with the Name field described in “Creating a Deployment Set from Runtime Assets” on page 32 .
srcAlias	The repository alias for the component. This attribute corresponds with the Name field described in “Connecting to a Source Repository” on page 69 .

The <Composite> tag describes composites that are part of the deployment set. You can set the following attributes for the <Composite> tag.

Attribute	Description
name	<p>Name of composite from the ACDL. You can use an asterisk (*) as a wildcard character. For example, if you set name to “Deploy*”, Project Automator adds all composites with a name beginning with “Deploy” to the deployment set.</p> <p>To use a wildcard character to return composites that contain an asterisk (*) in the name, you must add an escape character (\) before the asterisk in name (example, *). For example, to add all composites with the name “project*”, you would set</p>

Attribute	Description
srcAlias	name to "project*". To add all composites with the name "*project", you would set name to "*project".
type	<p>Runtime type of the asset. Set to:</p> <ul style="list-style-type: none"> ■ <i>MFT</i> ■ <i>APIGateway</i> ■ <i>AgileApps</i> ■ <i>ApplicationPlatform</i> ■ <i>BPM</i> ■ <i>Broker</i> ■ <i>DES</i> ■ <i>EDA</i> ■ <i>EventServer</i> <div style="border: 1px solid gray; background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> </div> <ul style="list-style-type: none"> ■ <i>IS</i> ■ <i>MWS</i> ■ <i>Optimize</i> ■ <i>RULES</i> ■ <i>TN</i> ■ <i>UniversalMessaging</i> <p>You can use an asterisk (*) as a wildcard character. For example, if you set type to "*", Project Automator adds composites from all runtimes to the deployment set.</p>

The <Component> tag describes the assets that are part of the deployment set. You can set the following attributes for the <Component> tag.

Attribute	Description
componentType	<p>Type of asset. For example, isdocumenttype.</p> <p>You can use an asterisk (*) as a wildcard character. For example, if the componentType attribute is set to "*", Project Automator adds all asset types to the deployment set.</p>
compositeName	<p>Name of composite in which the asset is located. You can use an asterisk (*) as a wildcard character. For example, if the compositeName attribute is set to "is*", Project Automator adds all assets with a composite name beginning with "is" to the deployment set.</p> <p>To use a wildcard character to return all composites that contain an asterisk (*) in the name, you must add an escape character (\) before the asterisk in compositeName (example, *). For example, to add assets from composites with the name "project*", you would set compositeName to "project*". To add all composites with the name "*project", you would set compositeName to "*project*".</p>
name	<p>Name of the asset. You can use an asterisk (*) as a wildcard character. For example, if the name attribute is set to "is*", Project Automator adds all assets with a name beginning with "is" to the deployment set.</p> <p>To use a wildcard character to return all assets that contain an asterisk (*) in the name, you must add an escape character (\) before the asterisk in name (example, *). For example, to add all assets with the name "project*", you would set name to "project*". To add all assets with the name "*project", you would set name to "*project*".</p>
srcAlias	Alias name of the host server.
type	<p>Runtime type of the asset. Set to:</p> <ul style="list-style-type: none"> ■ <i>MFT</i> ■ <i>APIGateway</i> ■ <i>BPM</i> ■ <i>Broker</i>

Attribute	Description
	<ul style="list-style-type: none"> ■ <i>DES</i> ■ <i>EDA</i> ■ <i>EventServer</i> <div style="background-color: #f0f0f0; padding: 5px; margin: 5px 0;"> <p>Note: Deployer supports deployment of assets to Event Servers of version 9.5 or earlier only.</p> </div> <ul style="list-style-type: none"> ■ <i>IS</i> ■ <i>MWS</i> ■ <i>Optimize</i> ■ <i>RULES</i> ■ <i>TN</i> ■ <i>UniversalMessaging</i> <p>You can use an asterisk (*) as a wildcard character. For example, if you set type to "*", Project Automator adds assets from all runtimes to the deployment set.</p>

Defining a Deletion Set for Repository-Based Deployment

You define deletion sets for repository-based deployment as follows:

```
<DeletionSet
autoResolve="full" description="description" name="deletion_set">

  <Component componentType="component_type" compositeName="composite_name"
name="name" srcAlias="repository_alias" type="runtime_type"/>

</DeletionSet>
```

Note:

Application Platform has different assets when deploying from a repository and when defining a deletion set. Deletion set assets represent the entire parent composite. This is because Application Platform cannot delete individual assets. Therefore you must set name=compositeName and type=compositeType. Thus there are only three types of deletion-time assets: "bundle", "war", and "properties".

The following table describes the attributes in the <DeletionSet> tag:

Note:

For a description of the attributes for the <Component> tag, see [“Defining a Deployment Set for Repository-Based Deployment” on page 313](#).

Attribute	Description
autoResolve	The autoResolve attribute is optional, but if you supply a value, set it to <i>full</i> . If autoResolve is set to <i>full</i> , Deployer automatically finds dependent assets and adds them to the deletion set. If autoResolve is not supplied and Deployer finds dependent assets, Deployer creates the deletion set with unresolved dependencies and deployment will fail. For more information about the autoResolve attribute, see “Defining a Deployment Set for Repository-Based Deployment” on page 313 .
description	Description for the deletion set. This attribute corresponds with the Description field described in “Creating a Deletion Set” on page 47 .
name	Name to use for the deletion set. This attribute corresponds with the Name field described in “Creating a Deletion Set” on page 47 .

Building a Project for Runtime-Based Deployment

You define a build for a runtime-based deployment project as follows:

```
<DeploymentBuild
name="build name" description="build description">/DeploymentBuild
```

Note:

If the project already has a build with the same name, Deployer overwrites it.

The following table describes the attributes in the <DeploymentBuild> tag:

Attribute	Description
name	Name of the build. This attribute corresponds to the Name field described in “Creating a Build” on page 51 .
description	Description of the build. This attribute corresponds to the Description field described in “Creating a Build” on page 51 .

Mapping a Project

You define a map to create for a project as follows:

```
<DeploymentMap
```

```

name="map name" description="map description"></DeploymentMap>
<MapSetMapping mapname="map name" setName="name of deployment or
deletion set">
  <alias>target
server alias</alias>
  <alias>target
server alias</alias>
  <group>target
group</group>
  <group>target
group</group>
</MapSetMapping>

```

Note:

If the project already has a map with the same name, Deployer overwrites it.

The following table describes the attributes in the <DeploymentMap> tag:

Attribute	Description
name	Name of the deployment map. This attribute corresponds to the Name field described in “Mapping a Project to Target Servers and Target Groups” on page 225.
description	Description of the deployment map. This attribute corresponds to the Description field described in “Mapping a Project to Target Servers and Target Groups” on page 225.

Mapping a Runtime-Based Project

You define a deployment map for a runtime-based project with the <DeploymentMap> and <MapSetMapping> tags as follows:

```

<DeploymentMap
name="map name" description="map description"></DeploymentMap>

<MapSetMapping mapname="map name" setName="name of deployment or
deletion set">
  <alias>target_server_alias</alias>

  <group>target_group</group>

  <group>target_group</group>

</MapSetMapping>

```

Note:

If the project already has a map with the same name, Deployer overwrites it.

The following table describes the attributes in the <DeploymentMap> tag. Each attribute corresponds to a field in the GUI as described in [“Mapping a Project to Target Servers and Target Groups”](#) on page 225.

Attribute	Description
name	Name of the deployment map. This attribute corresponds to the Name field.
description	Description of the deployment map. This attribute corresponds to the Description field.

The <MapSetMapping> tag uses the following attributes to define the aliases contained in each deployment map:

Attribute	Description
mapname	Name of the deployment map. This attribute should match the name attribute of the <DeploymentMap> tag.
setName	Name of the deployment or deletion set. <ul style="list-style-type: none"> ■ For deployment sets, this attribute should match the name attribute of the <DeploymentSet> tag as described in “Defining Deployment and Deletion Sets for Runtime-Based Deployment” on page 310 or “Defining a Deployment Set for Repository-Based Deployment” on page 313. ■ For deletion sets, this attribute should match the name attribute of the <DeletionSet> tag as described in “Defining Deployment and Deletion Sets for Runtime-Based Deployment” on page 310 or “Defining a Deletion Set for Repository-Based Deployment” on page 317.
alias	The server alias of the target server. This attribute corresponds to the target server you select when you click Add Target Server as described in “Mapping a Project to Target Servers and Target Groups” on page 225.
group	The alias of the target group. This attribute corresponds to the target group you select when you click Add Target Group as described in “Mapping a Project to Target Servers and Target Groups” on page 225.

Mapping a Repository-Based Project

You define a deployment map for a repository-based project with the <DeploymentMap> and <MapSetMapping> tags as follows:

```
<DeploymentMap
name="map name" description="map description"></DeploymentMap>

<MapSetMapping mapname="map name"setName="name of deployment or
```



```

deletion set">
  <alias type="alias_type">target_server_alias</alias>

  <alias logicalServer="logical_server" type="alias_type">target server alias
</alias>
</MapSetMapping>

```

Note:

If the project already has a map with the same name, Deployer overwrites it.

The following table describes the attributes in the `<DeploymentMap>` tag. Each attribute corresponds to a field in the GUI as described in [“Mapping a Project to Target Servers and Target Groups” on page 225](#).

Attribute	Description
name	Name of the deployment map. This attribute corresponds to the Name field.
description	Description of the deployment map. This attribute corresponds to the Description field.

The `<MapSetMapping>` tag uses the following attributes to define the aliases contained in each deployment map:

Attribute	Description
mapname	Name of the deployment map. This attribute should match the name attribute of the <code><DeploymentMap></code> tag.
setName	Name of the deployment or deletion set. <ul style="list-style-type: none"> ■ For deployment sets, this attribute should match the name attribute of the <code><DeploymentSet></code> tag as described in “Defining Deployment and Deletion Sets for Runtime-Based Deployment” on page 310 or “Defining a Deployment Set for Repository-Based Deployment” on page 313. ■ For deletion sets, this attribute should match the name attribute of the <code><DeletionSet></code> tag as described in “Defining Deployment and Deletion Sets for Runtime-Based Deployment” on page 310 or “Defining a Deletion Set for Repository-Based Deployment” on page 317.
alias type	The runtime type of the target server alias. This attribute corresponds to the Select Server list described in “Mapping a Project to Target Servers and Target Groups” on page 225 .

Attribute	Description
alias logicalServer	(BPM assets only.) The logical server alias of the target server. This attribute corresponds to the Physical Server field as described in “Connecting to BPM Process Model Servers” on page 26.
alias	The server alias of the target server. This attribute corresponds to the target server you select when you click Add Target Server as described in “Mapping a Project to Target Servers and Target Groups” on page 225.
group type	The runtime type of the target group. This attribute corresponds to the Select Server list described in “Mapping a Project to Target Servers and Target Groups” on page 225.
group logicalServer	(BPM assets only.) The logical server to which the target group is mapped. This attribute corresponds to the target server or target group you select after you click Add Target Group as described in “Mapping a Project to Target Servers and Target Groups” on page 225.
group	The group mapping for assets that are not targeted to a logical server. This attribute corresponds to the target group you select when you click Add Target Group as described in “Mapping a Project to Target Servers and Target Groups” on page 225.

Creating a Deployment Candidate for Runtime-Based Deployment

You define a deployment candidate for a runtime-based project with the `<DeploymentCandidate>` tag as follows:

```
<DeploymentCandidate
name="candidate
name" description="candidate description"
buildName="build
to use" mapName="map to use">/DeploymentCandidate>
```

Note:

If the project already has a deployment candidate with the same name, Deployer overwrites it.

The following table describes the attributes in the `<DeploymentCandidate>` tag when creating a deployment candidate for runtime-based deployment. Each attribute corresponds to a field in the GUI as described in [“Deploying a Project”](#) on page 234.

Attribute	Description
name	Name of the deployment candidate. This attribute corresponds to the Name field.
description	Description of the deployment candidate. This attribute corresponds to the Description field.
buildName	(Runtime-based deployment only.) Name of the project build to deploy. This attribute corresponds to the Project Build field.
mapName	Name of the deployment map that identifies the target servers to which to deploy the assets. This attribute corresponds to the Deployment Map field.

Creating a Deployment Candidate for Repository-Based Deployment

You define a deployment candidate for a repository-based project with the `<DeploymentCandidate>` tag as follows:

```
<DeploymentCandidate
name="candidate
name" description="candidate description"
mapName="map to
use">/DeploymentCandidate>
```

Note:

If the project already has a deployment candidate with the same name, Deployer overwrites it.

The following table describes the attributes in the `<DeploymentCandidate>` tag when creating a deployment candidate for repository-based deployment. Each attribute corresponds to a field in the GUI as described in [“Deploying a Project” on page 234](#).

Attribute	Description
name	Name of the deployment candidate. This attribute corresponds to the Name field.
description	Description of the deployment candidate. This attribute corresponds to the Description field.
mapName	Name of the deployment map that identifies the target servers to which to deploy the assets. This attribute corresponds to the Deployment Map field.

Running Project Automator

Go to the *Integration Server_directory/instances/instance_name/packages/WmDeployer/bin* directory and run this command:

```
{projectAutomator.bat|projectAutomatorUnix.sh|projectAutomatorMac.sh}  
full_path_to_XML_file
```

To configure secure ports for the servers, run the command with the following command options:

```
{projectAutomator.bat|projectAutomatorUnix.sh|projectAutomatorMac.sh}  
full_path_to_XML_file -useSSL [-senderCert path_to_cert] [-privKey path_to_key] [-caCert  
path_to_cert] [-useJSSE]
```

where

- `-useSSL` indicates that you want to connect to the server over an HTTPS port.
- `-senderCert path_to_cert` Optional. Specifies the location of the client certificate for Project Automator.
- `-privKey path_to_key` Optional. specifies the location of the private key for the certificate.
- `-caCert path_to_cert` Optional. Specifies the location of the signing authority's certificate. If the certificates and private key do not exactly match the ones configured for Project Automator in the Integration Server installation, the command will fail.
- `-useJSSE` Required only for JSSE. Indicates to connect over the HTTPS port with Java Secure Socket Extension (JSSE) enabled. If an HTTPS port makes use of JSSE, then pass `-useJSSE` along with the `-useSSL` option.

Ensure that the certificates are in DER format; if not, convert them to DER format using a certificate management tool, such as Java `keytool`. For more information on the supported types of certificate format, see the *Prerequisites to Configuring a Port for SSL* section in the *webMethods Integration Server Administrator's Guide* .

Before executing the specified XML file, Project Automator validates the XML using one of the following schemas located in the *Integration Server_directory/instances/instance_name/packages/WmDeployer/config* directory:

- `ProjectAutomatorForRepository.xsd`
- `ProjectAutomatorForRuntime.xsd`
- `ProjectAutomator.xsd`

11 Assets with a Specific Deployment Setup

- Deploying Process Models with E-Forms 326
- Deploying Optimize Assets 327

Deploying Process Models with E-Forms

Note:

These instructions apply to runtime-based deployment only. For repository-based deployment, if you want to deploy ProcessModels with e-forms, you must configure the e-form listener manually in the target environment. For more information about configuring the e-form listener manually, see *Implementing E-form Support for BPM* (for My webMethods Server).

If you are creating a ProcessModel deployment set, the e-forms might trigger process steps in the process models. In this case, follow these instructions:

1. When you define the ProcessModel deployment set, JCR or CSP files will appear as dependencies. You must include them in the ProcessModel deployment set unless they already exist on the target server and they already specify the correct paths to the e-form templates and e-form instances folders. For instructions on resolving dependencies, see [“Resolving Dependencies” on page 45](#).

Note:

My webMethods Server assets (that is, the e-form templates and e-form instances folders) do not appear as dependencies.

2. When you map the ProcessModel deployment set to the target servers, substitute the password configuration values for the JCR or CSP files. These passwords, which Process Engines use to connect to the My webMethods Server or Content Service Platform that hosts e-forms, must be correct for the target environment. Also change other JCR or CSP file configuration values to be correct for the target environment if necessary.

Note:

Configure the CSP in the target server with the same structure and content as the source server.

3. If you are using e-forms with My webMethods Server, perform the following additional tasks:
 - a. Define an MWS deployment set that contains the e-form templates and e-form instances folders from the My webMethods Server that hosts e-forms in the source environment. In the project settings for the project that contains this deployment set, set the **Export Content (Documents)** property to **Yes** (see the **Export Content (Documents)** property as described in [“Creating a Project” on page 209](#)).

Note:

This setting is set to **No** by default. Since project settings affect all deployment sets in the project, make sure that other MWS deployment sets you include in the project can share this setting, or do not include other MWS deployment sets in the project.

- b. Before deploying, go to the My webMethods Server that hosts e-forms in the source environment and delete the contents of the e-form instances folder.

Deploying Optimize Assets

When you use Deployer to deploy Optimize assets, the process often requires modifications to the Analytic Engine DB schema on the target server. The modifications result from the automatic execution of Data Definition Language (DDL) statements on the database. You can configure the Analytic Engine to run in Static DB Schema mode, where the automatic execution of DDL statements is disabled. For more information, see [“Disabling Automatic Execution of DDL Statements” on page 327](#).

When the Analytic Engine runs in Static DB Schema mode, the deployment process for Optimize assets runs through two stages. This ensures that no unauthorized modification of the database schema is allowed.

- **First stage (DDL Recording)** - during this stage, the deployment process carries out the following actions:
 1. Old versions of the Optimize assets are removed from the Analytic Engine cache.
 2. Data Manipulation Language (DML) statements are executed.
 3. DDL statements are collected but not executed.
- After these procedures finish, you manually execute the Analytic Engine SQL scripts.
- **Second stage (Actual Deployment)** - during this stage, the deployment process carries out the following actions:
 1. Determines if the database has been updated with the SQL scripts as described in the first stage.
 2. Creates the deployed Optimize assets.

Note:

In practice, these stages are part of a single procedure, as described in [“Deploying Optimize Assets in Static DB Schema Mode” on page 328](#). However, be aware that you must complete both deployment stages for each Optimize deployment set before you start deploying another set. Otherwise you will create orphaned objects in your database.

Disabling Automatic Execution of DDL Statements

You must disable the automatic execution of DDL statements to be able to safely deploy Optimize deployment sets as described in [“Deploying Optimize Assets in Static DB Schema Mode” on page 328](#).

➤ To disable the automatic execution of DDL statements

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**.

2. On the Define Environments page, click the name of the environment with which you want to work.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Database Settings**.
5. In the **Database Settings for Analytic Engine** area, click **Disable DDL Statements** to disable the automatic execution of DDL statements.
6. Click **Save** to save changes.

Note:

If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. Click **Deploy All** to deploy all configuration files to all logical servers in an environment, or click **Deploy Updates** to deploy only the modified configuration files to the affected logical servers in the environment.

The status of the deployment operation appears after the operation is completed. The status includes a list of the files that were deployed to the environment and also lists any errors that occurred.

Note:



If you edit a configuration and deploy only the updates, the logical servers in the environment must be restarted for the new configuration settings to take effect.

Deploying Optimize Assets in Static DB Schema Mode

You deploy Optimize assets by adopting the following procedure only when Analytic Engine runs in Static DB Schema mode. For deploying Optimize assets when Static DB Schema mode is not activated, see [“Deploying a Project” on page 234](#).


➤ To deploy Optimize assets when Analytic Engine runs in Static DB Schema mode

1. Make sure you have deployed an environment with Analytic Engine running in Static DB Schema mode as described in [“Disabling Automatic Execution of DDL Statements” on page 327](#).
2. In Integration Server: **Solutions > Deployer**.
3. On the **Deployer > Settings** page, in the **General Deployment Defaults** area, make sure the **Batch Size** property is set to 0 to avoid batching of the deployment set.

4. Go to the **Deployer > Projects** page and configure an Optimize deployment candidate as described in “[Creating a Project](#)” on page 209.
5. Click  **Deploy**. Deployer displays the **Projects > project > Deploy** page and lists all deployment candidates that exist for the selected project.
6. To deploy the project, click  in the **Deploy** column for the deployment candidate.

Deployer does the following:

- Removes older versions of the assets in the project build from the target servers to prepare the servers for deployment.
 - Creates a deployment report and lists the report in the **Deployment History** area. The deployment report contains the location of the .sql file that you need to execute manually before continuing with the second deployment phase.
7. Locate and execute the SQL scripts.

You must have database administrator privileges on the database to be able to execute the scripts. You will find the script(s) that must be executed in a .sql file on the Analytic Engine host file system. The exact location is specified in the deployment report in the **Deployment History** area.
 8. Click  in the **Deploy** column to complete the deployment.

Optimize Deployment Usage Notes

The purpose of this topic is to help you understand various important points concerning the deployment of Optimize assets in Static DB Schema mode.

There are two prerequisites critical to the successful deployment of Optimize assets to a target Analytic Engine running in Static DB Schema mode:

1. For each deployment set you want to deploy, you must always execute the following operations in this order:
 - a. Execute the DDL recording stage.
 - b. Execute all generated SQL statements.
 - c. Execute the actual deployment stage.
2. Ensure that each asset included in the process is removed or deployed, instead of updated.

The Analytic Engine attempts to remove all assets within each deployment set in reverse order and then attempts to deploy them in the specified order. An asset cannot be removed if there are other assets depending on it.

Here is the Optimize assets dependency graph:

```
ILink - Dimension { / Hierarchy \  
                  \ EventMap /      /      \      DataFilter /
```

If the Analytic Engine does not run in Static DB Schema mode and some assets cannot be removed, Deployer attempts to update these assets at deployment time. This is not supported by the Static DB Schema mode.

Potential Problems

When you deploy Optimize assets, it is possible to overlook settings, configure various settings incorrectly, or attempt unsupported actions. As a result, the deployment fails or other issues occur. This topic covers commonly encountered problems.

Deployer Batch Size

Deployer **Batch Size** must always be 0 (zero). If the batch size is set to any other value, Deployer will try to split your deployment set into smaller chunks, which will violate both the prerequisites.

For more information about setting up the Batch Size property, see [“Settings for Runtime-Based Deployment Projects” on page 202](#) or [“Settings for Repository Based Deployment Projects” on page 207](#).

Removing Assets from a Deployment Set

If you decide that you no longer need an asset that has been previously deployed, you cannot simply remove the asset from your previously deployed deployment set and re-deploy it. If you do so, the existing asset on the target Analytic Engine (which is no longer in the deployment set) will prevent the removal of any assets it depends on.

To address this, you must first remove all assets from the target Analytic Engine that are depended on by the asset you want to remove from your deployment set. To do so:

1. Set up your target host as a source and create a new project with a deployment set containing all the assets you want to remove.
2. Set up your target host as a target again and map the deployment set created in step 1 to it.

Note:

For example, you might have two Optimize servers configured with different names and the same host and port.

3. After you create a deployment candidate, click **Create Checkpoint** to activate the **Rollback** button.
4. Click **Rollback** to remove all assets in the deployment set created in step 1 from the target Analytic Engine.

After all depended-on assets are removed, you can deploy the modified deployment set from your original source.

Two or More Deployment Sets for the Same Analytic Engine Using One Deployment Map

This is not supported. You can carry out either of the following supported methods as an alternative:

- Split the Optimize deployment sets into separate projects and complete both stages consecutively for each project.

or

- Remove the target Analytic Engine for all deployment sets. For each deployment set with no target specified in the deployment map, do the following:
 1. Add a target Analytic Engine in the deployment map.
 2. Execute both deployment sets and the generated DDLs in the correct order, as described in Step 1 in [“Optimize Deployment Usage Notes” on page 329](#).
 3. Remove the target Analytic Engine from the deployment map.

Executing DDL Statements for Two or More Analytic Engines

This situation applies when you have executed the DDL recording stage for one deployment set, and then attempt to execute the DDL recording stage for a second deployment set, with both deployment sets targeting the same Analytic Engine.

This is not supported. You must execute both deployment stages and the generated DDLs for each deployment set in the correct order, as described in Step 1 in [“Optimize Deployment Usage Notes” on page 329](#).

