EMC® Documentum® Composer

Version 6.5

Quick Start Guide

P/N 300-007-503-A02

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Published July 2008

Table of Contents

List of Figures

List of Tables

Chapter 1

What is Composer?

Composer is the next generation development tool for developing, building, and installing Documentum artifacts which are organized into projects. Most of the DAB/DAI functionality is available in Composer but there are several fundamental differences in concepts and terminology. The most important difference is that unlike DAB, Composer does not require a repository connection in order to view or modify application artifacts. You only need to connect to a repository when you want to install your projects or import existing objects.

Composer is shipped in two versions, the UI-based version with wizards, dialogs, and editors, and a non-UI command line version that includes a set of Ant tasks for each of the common Composer functions, such as import, build, and install. The non-UI version of Composer is also referred to as headless Composer.

Workspaces, Artifacts, and DARs

Composer is based on the Eclipse Integrated Development Environment (IDE) and thus adopted a similar terminology to comply with the Eclipse guidelines. Table 1, page 7 describes some of the new terminology that is used in Composer and relates it to the terminology that was used in DAB.

Table 1. Composer/DAB Comparison

Composer Terminology	Description	DAB Terminology
Artifact	A component of an application, such as an alias set, lifecycle, or method, packaged into a project.	Repository object
Project	A project contains the components (artifacts) that make up an application. A project resides on the local machine.	DocApp

Composer Terminology	Description	DAB Terminology
Workspace	A directory on the local machine which contains the project's artifact files. A project must be imported into a workspace before you can build, update, or install it. As a best practice you should set up at least three separate workspaces, as follows: • One workspace for UI-based Composer projects. • One workspace for importing and build projects using headless Composer. • One workspace for installing DARs using headless Composer.	Repository
DAR	This file is the installable binary file that is created by building a project. The file cannot be modified directly.	DocApp Archive

UI-based Composer

The Composer GUI should be used for all development tasks, such as developing an application from scratch or modifying an existing project. You also need to use the Composer GUI for importing 5.3 DocApps and DocApp archives.

Note: You cannot import a DAR file into Composer. A DAR file is similar to an executable file. It contains only the binary code and no source files.

Headless Composer

Headless Composer is most useful when you do not want to make any changes to a project but want to automate compiling an existing project and installing the DAR into a repository.

Note: Headless Composer cannot be used to create new projects or new artifacts.

UI-based Composer or Headless Composer?

Composer is available in two separate packages, Composer and headless Composer. Composer has a user interface, headless Composer is run from the command line using a batch file and Ant tasks. Table 2, page 9 describes the main differences.

Table 2. UI-based and Headless Composer Comparison

Features/Functionality	UI-based Composer	Headless Composer
Import DocApps from repository	Yes	No
Import DocApp archives	Yes	No
Import project from local directory	Yes	Yes
Import Artifact from repository	Yes	No
Build project	Yes	Yes
Install project	Yes	Yes
Import DAR	No	No
Install DAR file	No	Yes
	The Composer UI lets you install the project, a process that includes automatically generating and installing a DAR file “behind the scenes”. However, there is no separate “Install DAR File” option in the Composer UI.	Headless Composer lets you install a DAR file using the **emc.install** Ant task.

Chapter 2

Obtaining, Installing, and Configuring UI- based Composer

Composer is delivered in the form of a compressed .Zip file that contains the Eclipse platform and all required plug-ins.

To set up UI-based Composer:

- Download Composer package and extract the package to a local directory.
- Download and install Java JDK 1.5 (if it is not already installed on your local machine).
- Set JAVA_HOME environment variable to point to Java JDK 1.5.
- Configure the connection broker.
- Start Composer and create a workspace.

For more detailed information on how to install and configure Composer, see Chapter 1 in the *Composer User Guide*.

Using Composer

After you have installed Composer, you are ready to work on Documentum projects. You can create a project from scratch, migrate a 5.3 DocApp or DocApp archive into Composer, or import an existing project from a local directory. Figure 1, page 11 describes a typical workflow in Composer.

Figure 1. Composer Workflow

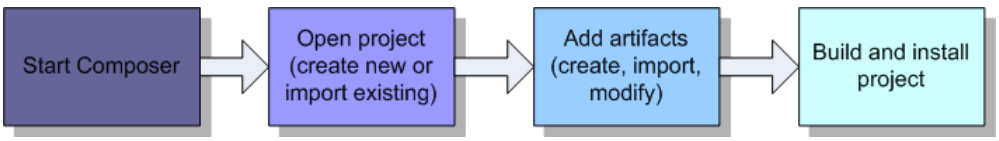


Table 3, page 12 describes the most common Composer tasks.

Table 3. Composer Tasks

Task	Steps	User Guide Topic
Create a project from scratch	Select **File** > **New** > **Project** > **Documentum Project** > **Documentum Project**	Chapter 2, "Creating a new project"
Add artifacts to project	• Expand **Artifacts** folder in project. • Right-click associated artifact folder, select **New** > **Other** and select artifact from list	Chapter 2, "Adding or selecting artifacts"
Import artifact from repository into project	Select **File** > **Import** > **Documentum** > **Artifacts From Repository**	Chapter 2, "Importing artifacts"
Import an existing project from local directory	Select **File** > **Import** > **Documentum** > **External Projects Into Workspace**	Chapter 2, "Importing a project"
Migrate a DocApp	Select **File** > **New** > **Project** > **Documentum Project** > **New Documentum Project from DocApp**	Chapter 3, "Migrating a repository DocApp"
Migrate a DocApp archive	Select **File** > **New** > **Project** > **Documentum Project** > **Documentum Project from Local DocApp Archive**	Chapter 3, "Migrating a DocApp archive"

Chapter 3

Obtaining and Installing Headless Composer

Headless Composer is delivered in the form of a compressed .Zip file that contains headless Eclipse and all required plug-ins.

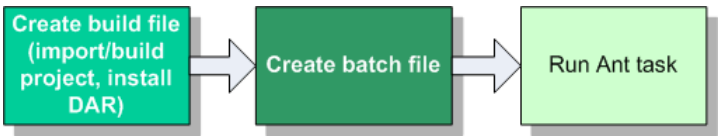
To set up headless Composer:

- Download headless Composer package for your platform and extract the package to a local directory. The headless Composer package has the format **DCTM_Headless_Composer_<platform>_<version>.zip**. When you extract headless Composer, do not extract it to a directory name that contains spaces. If the headless Composer directory path contains any spaces, Ant tasks do not run correctly.
- Configure the connection broker.
- Create at least two separate workspaces, one for importing and building project, and one for installing DAR files into a repository.

Using Headless Composer

After you have installed headless Composer, you need to create your build and batch files to run Ant tasks. Figure 2, page 13 describes a typical workflow when using headless Composer.

Figure 2. Headless Composer Workflow



In your build scripts and batch file you should:

- Create a clean build workspace.
- Copy the projects into the build workspace.

- Import projects into the clean build workspace by calling the **emc.importProject** Ant task.
- Build projects and create DAR by calling **emc.build** and **emc.dar** Ant tasks.
- Create a clean install workspace.
- Install the DAR file using the previous build process by calling the **emc.install** Ant task.

Creating a Build File

The build file calls the Ant task you want to execute. The build file must be in XML format. Table 4, page 14 describes the Ant task that are supported by headless Composer.

For more information about parameters and syntax, see the Chapter 20 in the *Composer User Guide*.

For detailed example on how to use headless Composer to install a DAR file, see Chapter 4, Installing a DAR File.

Table 4. Headless Composer Ant Tasks

Ant Task	XML Code	Description

emc.
importProject
	`<emc.importProject`
	`dmproject="`***`project`***`"`
	`failonerror="`***`true/false`***`" />`
		Imports a project into the workspace.

emc.build	`<emc.build`
	`dmproject="`***`project`***`"`
	`failonerror="`***`true/false`***`" />`
		Compiles an existing Composer project.
		
		The project should be compiled in a clean workspace.

emc.dar	`<emc.dar`
	`dmproject="`***`project`***`"`
	`manifest="bin/dar/`***`project`***`.dardef.`
	`artifact"`
	`dar="c:/`***`project`***`.dar" />`
		Generates a DAR file from an existing project.

Ant Task	XML Code	Description
emc.install	<emc.install dar="C:/***project***.dar" docbase="***repository***" username="***name***" password="***password***" domain="" />	Installs a DAR file into a repository.

Creating a Batch File

The batch file sets up the environment variables and passes the build file, as described in Table 5, page 15.

For a detailed example batch file, see Chapter 4, Installing a DAR File.

Table 5. Batch File

Code	Description
echo off setlocal	
set ECLIPSE="C:\<..>\ComposerHeadless"	Specifies the location of headless Composer. The directory name should not contain any spaces.
set BUILDWORKSPACE="c:\<..>\ ComposerHeadless\<*build_workspace*>"	Specifies the workspace that is used when a project is build.
set BUILDFILE="R:\<*path*>\<*buildfile*>.xml"	Specifies the path and the name of the build file that contains the required parameters for building a project.
set INSTALLFILE="R:\<*path*>\<*installfile*>.xml"	Specifies the path and the name of the file that contains the required parameters for installing a DAR file.
set DARPROJECTSDIR="R:\<*path*>\<*directory*>"	Specifies the directory where the generated DAR file is stored.
set INSTALLWORKSPACE="c:\<..>\ComposerHeadless\ <*install_workspace*>"	Specifies the workspace that is used when a DAR file is installed.

Code	Description

`rmdir /S /Q %BUILDWORKSPACE%`
`rmdir /S /Q %INSTALLWORKSPACE%`
`xcopy %DARPROJECTSDIR% %BUILDWORKSPACE% /E /I`
	Removes all directories and files in the build and install workspaces and copies the project files into the build workspace and DAR file into the install workspace.

`java -cp %ECLIPSE%\startup.jar org.eclipse.`
`core.launcher.Main`
`-data %BUILDWORKSPACE%`
`-application org.eclipse.ant.core.antRunner`
`-buildfile %BUILDFILE%`
	Sets the Java parameters for the build workspace and build file. All parameters must be entered on the same line without any line breaks.

`java -cp %ECLIPSE%\startup.jar org.eclipse.`
`core.launcher.Main`
`-data %INSTALLWORKSPACE%`
`-application org.eclipse.ant.core.antRunner`
`-buildfile %INSTALLFILE%`
	Sets the Java parameters for the installation workspace and installation file. All parameters must be entered on the same line without any line breaks.

Running Ant Tasks

After you have created your batch and build file, you can run the Ant tasks, as follows:

```
C:\><..>\ComposerHeadless <batch file>.bat <build file>.xml
```

Where <batch file> is the name of the batch file and <build file> is the name of the build file.

Chapter 4

Installing a DAR File

You can install a DAR file with Composer or the DAR Installer program.

Installing a DAR file with headless Composer

One of the most common tasks for headless Composer is to import a project, build it, and then install the DAR file into a repository. The following procedure describes this process and contains example code for the associated build and batch files.

To install a DAR file:

1. Create the build file that imports the project, builds the project, and generates the DAR file using the **emc.importProject**, **emc.build**, and **emc.dar** Ant tasks.

   The following **DARBuild.xml** example build file contains the code to import and build the **SmartContainer** project, and to generate the **SmartContainer.dar**.

```
<?xml version="1.0"?>
<project
name="build-smart-container"
default="build">
<target name="build">

<emc.importProject
dmproject="SmartContainer"
failonerror="true"/>
```

XML declarations and tags.

Imports the **SmartContainer** project.

The **failonerror** parameter specifies that the import fails if any errors are encountered during the import process.

```
<emc.build
dmproject="SmartContainer"
failonerror="true"/>
```

Builds the **SmartContainer** project.

The **failonerror** parameter specifies that the build fails if any errors are encountered during the build process.

```
<emc.dar
dmproject="SmartContainer"
manifest="bin/dar/
smartcontainer.dardef.artifact"
dar="c:/ComposerHeadless/
build_workspace/SmartContainer/
SmartContainer.dar"/>
```

Generates the **SmartContainer.dar** file from the **SmartContainer** project.

The **manifest** parameter specifies the location of the manifest file, which contains the packaging instructions for the DAR file.

The **dar** parameter specifies the location where generated the .dar file is stored.

```
</target>
</project>
```

XML closing tags.

2. Create the build file that installs the DAR file into the repository.

   The following example **DARInstall.xml** build file contains the code to install the **SmartContainer.dar**.

```
<?xml version="1.0"?>
<project name="install-smart-container"
default="install">
<target name="install">
```

XML declarations and tags.

```
<emc.install
dar="c:/ComposerHeadless/build_workspace/
SmartContainer/SmartContainer.dar"
docbase="repo1"
username="user"
```

Installs the **SmartContainer.dar** file in the **repo1** repository.

```
password="aabbcc"
domain="" />

</target>
</project>
```

XML closing tags.

3. Create the batch file that calls both build files.

   The following **SmartContainerBuild.bat** file sets up the environment variables, workspaces, and **DARBuild.xml** and **DARInstall.xml** files.

```
echo off setlocal

set ECLIPSE="C:\ComposerHeadless"

set BUILDWORKSPACE="C:\ComposerHeadless\
build_workspace"

set BUILDFILE="R:\Smart_Container\Main\buildEnv\
DARBuild.xml"

set INSTALLFILE="R:\Smart_Container\Main\
buildEnv\DARInstall.xml"

set DARPROJECTSDIR="R:\Smart_Container\Main\
darProjects"

set INSTALLWORKSPACE="C:\ComposerHeadless\
install_workspace"

rmdir /S /Q %BUILDWORKSPACE%
rmdir /S /Q %INSTALLWORKSPACE%
xcopy %DARPROJECTSDIR% %BUILDWORKSPACE% /E /I
```

Sets the headless Composer workspace.

Sets the workspace that is used when building the project.

Sets the build file that is used for building the project.

Sets the installation build file that is used for installing the DAR file.

Sets the DAR file location directory. This is where the DAR file is stored when it is generated.

Sets the workspace that is used during the installation process.

Cleans the build workspace and install workspace, and copies the content from the DAR file directory to the build workspace.

```
java -cp %ECLIPSE%\startup.jar org.eclipse.
core.launcher.Main
-data %BUILDWORKSPACE%
-application org.eclipse.ant.core.antRunner
-buildfile %BUILDFILE%
```

Sets the Java parameters for building the project and generating the DAR file.

```
java -cp %ECLIPSE%\startup.jar org.eclipse.
core.launcher.Main
-data %INSTALLWORKSPACE%
-application org.eclipse.ant.core.antRunner
-buildfile %INSTALLFILE%
```

Set the Java parameters for installing the DAR file.

4. Build the project and generate the DAR file by running the Ant task, as follows:

   ```
   C:\ComposerHeadless> SmartContainerBuild.bat DARBuild.xml
   ```

5. Install the DAR file by running the Ant task, as follows:

   ```
   C:\ComposerHeadless> SmartContainerBuild.bat DARInstall.xml
   ```

Installing a DAR file with the DAR Installer

You can use the DAR Installer program to install a DAR file to a repository if you do not want to use Composer or do not have it available. The DAR Installer program is useful in cases where you want to de-couple the development of DAR files from the installation of DAR files. When you open the DAR Installer program, it creates three folders in your Composer installation directory:

- darinstallerconfig - contains configuration files for the DAR Installer program
- darinstallerlogs - the default location of the log files
- darinstallerworkspaces - workspaces that are created and used by the DAR Installer program. The DAR Installer program does not delete these workspaces automatically, so you occasionally need to clean up this directory. The workspace directories are named in the following form: darinstaller_workspace_yyyy-mm-dd-hh-mm-ss.

The DAR Installer requires you to fill in certain values that are marked with an asterisk (*). All other fields are optional. For a description of the fields for the DAR Installer Utility, see . To install a DAR file:

To install a DAR file:

1. Download the DAR Installer zip file from the same place that you downloaded Composer. You can find the DAR Installer Plugin by going to https://emc.subscribenet.com/control/dctm/search and searching for “Composer” to reach the Documentum Composer download site.
2. Unzip the DAR Installer zip file to the root of your Composer or headless Composer installation directory.

3. Run darinstaller.exe, which is located in the Composer root directory, to start the DAR Installer Plugin.
4. In the **DAR Details** section, specify values for the fields.
5. In the **Connection Broker Details** section, specify values for **Connection Broker Host** and **Connection Broker Port** and click **Connect**.
6. In the **Repository Details** section, specify values for the fields and click **Install** to install the DAR file to the repository.

You can view the log for the DAR installation by selecting the log file from the **Log File** drop down menu and clicking **Open**.

Table 6. DAR Installer fields

Parameter	Required	Description
DAR	Yes	The absolute file path to the .dar file that you want to install. The file path cannot contain any I18N characters or the installation will fail.
Input File	No	The absolute file path to the install-based parameter file
Local Folder	No	The absolute file path to localized .properties files. If you want to make your application available in other languages, you need to localize the project data such as labels, tabs, and descriptions.
Log File	No	The file to save the log to. If this is not specified, the file defaults to <DAR>.log
Connection Broker Host	Yes	The address of the Connection Broker
Connection Broker Port	Yes	The port of the Connection Broker Repository
Repository	Yes	The name of the repository that you want to install the DAR file to. Click on the Connect button after entering the Docbroker host and port to retrieve the available repositories.

Parameter	Required	Description
User Name	Yes	The login name for the repository
Password	Yes	The password for logging into the repository
Domain	No	The domain where the repository resides