EMC® Documentum® Composer

Version 6.5 SP2

Introduction to Documentum Composer for DAB users

P/N 300-009-174 A01

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

Published June 2009

Table of Contents

List of Tables

Chapter 1

Introduction to Composer for DAB users

Documentum Composer is an Eclipse-based IDE that allows developers to create and install Documentum applications. It provides much of the functionality of DAB (Documentum Application Builder) and DAI (Documentum Application Installer) and is intended to replace these two products.

Important:

Composer is built on top of the Eclipse framework and is fundamentally different from DAB in how you develop, build, and deploy applications. If you are a DAB user, it is highly recommended that you read this document to eliminate confusion between how to do things in DAB compared to how to do them in Composer. Composer was designed with previous DAB users in mind to help ease the transition from DAB to Composer, but some concepts are different.

This document is intended to explain the differences between DAB and Composer, highlight the new features of Composer, and help previous DAB users understand the switch to Composer paradigms.

Terminology differences

Many key Composer terms and concepts are different than in DAB. Explaining all of the differences are out of the scope of this document. However, the following table highlights some of the major differences between DAB and Composer:

Table 1. Terminology differences

DAB term	Composer term
Docbase objects	Artifacts
DocApp	Composer Project
DocApp archive	DAR file
DAI	DAR Installer
Checkout of objects	Artifact import
Checkin of objects	Project installation

Artifacts

An artifact in Composer corresponds to a Docbase object in DAB. An artifact is typically a single entity such as a type or a lifecycle. You can import artifacts directly from the repository into a Composer project or create new artifacts and install them to the repository. Artifacts are installed as part of a Composer project. You currently cannot install a single artifact without installing the entire Composer project.

Composer projects

A Composer project corresponds to a DocApp in DAB. A Composer project is a collection of artifacts and other configuration files that describe the project. Composer projects are structured differently from DocApps. Most notably, Composer projects are created and edited offline, without a connection to a repository, while creating and editing DAB DocApps require a connection a repository.

In many cases, you can convert your existing version 5.3 or later DocApps to Composer projects, so you can modify them with Composer. For more information on converting your DocApps and DocApp archives to Composer projects, see the *Converting DocApps and DocApp archives to Composer projects* chapter in the *Composer User Guide*.

Composer DAR files

A Composer DAR file corresponds to a DocApp archive. A DAR file is a compact, installable archive of a Composer project. It does not contain any source files, so you cannot create a Composer project from a DAR file. You can use the DAR Installer that is provided with Composer to install DAR files or you can use headless Composer to install DAR files as part of an automated deployment environment.

Documentum products, such as Taskspace, have already converted from supplying DocApp archives to DAR files. For more information on packaging and deployment changes, see Chapter 3, Packaging and installing applications

DAR Installer

The DAR Installer provides similar functionality to DAI. It is a graphical user interface that is used to install DAR files. The DAR Installer is mainly used to install Documentum DAR files. It also allows you to decouple the development and deployment portions of your processes. For example, you can have developers submit their DAR files to an administrator with sufficient privileges to install the DAR files to a repository.

Artifact import

In DAB, you needed to check out objects from the repository before modifying them, which is no longer required in Composer. With Composer, you import artifacts from a repository into a project

and modify them offline. Modifying an artifact within Composer does not lock the artifact in the repository. Composer makes a connection to the repository during the actual import of the artifact, but does not maintain the connection when you are editing the artifacts. Creating new artifacts is also done offline within a project.

Project installation

In DAB, you needed to checkin objects to update them in the repository. In Composer, you install the project that contains the newly created artifacts or updated artifacts that you originally imported from a repository. You currently cannot install individual artifacts within a project. You must install your entire Composer project when you want to update a repository. Composer makes a connection to the repository during the installation of the project.

Chapter 2

Developing applications

A major difference between developing a DocApp and a Composer project is that DocApps required a connection to a repository and allowed you to modify Documentum artifacts in place, directly in the repository. With Composer, you create Documentum applications offline in the form of a Composer project.

Composer stores the Documentum artifacts locally until you are ready to deploy your project to a repository. When you are ready to deploy, Composer connects to the repository to install the project. This gives you many advantages such as not requiring a continuos connection to a repository and more granular source control capabilities.

Composer offers additional enhancements compared to DAB, including:

- A workspace that includes all of your projects in one view so you can work on multiple projects at one time
- Automatic building of your Composer project to catch errors before install time. This allows you to develop artifacts offline until they are ready for deployment
- More control over how artifacts are structured in Composer projects
- Integration with a Java IDE

Creating new Composer projects

DAB DocApps and Composer projects are similar in functionality and are the main representation of a Documentum application. Composer projects contain the artifacts that make up an application. The following procedure outlines the basic steps of creating Composer projects:

1. Launch Composer and create a new or use an existing workspace.
2. Create a new Documentum Composer project.
3. Create new artifacts or import existing artifacts from the repository.

Creating a Form or Process template

In DAB and in Composer, you create forms and process templates in the repository with Forms Builder or Business Process Builder. Once they are created, you must add them to your DocApp with

DAB or import them into your Composer project with Composer. The following procedure outlines the basic steps of creating the templates and importing them into your project with Composer:

1. Launch Forms Builder or Business Process Builder.
2. Create Form template or Process template in appropriate tool and save it.
3. Exit the tool.
4. Launch Composer.
5. Create a new project and import the form or business process artifact from the repository.

Creating BOF modules

One advantage of using Composer over DAB is that you can develop Java code for a BOF module in the same IDE as the artifacts. The following procedure outlines the basic steps of creating BOF modules in Composer:

1. Within a Composer Project, add source code for the BOF module to the src directory.
2. Compile and build the classes into a JAR file. The class files are in the bin directory of the Composer project.
3. Create a JAR definition artifact in Composer with the JAR file that you just created. Specify whether the JAR contains the interface, implementation or both. If you separated the interface JAR from the implementation JAR, you need to create two JAR definition artifacts.
4. Create a module artifact in Composer. Specify whether you want it to be a Standard Module, TBO, or SBO.
5. In the Core JARs section, specify the implementation JARs and the interface JARs.
6. Specify any other Java libraries that are needed to run the module.

Viewing and modifying XML application configuration files

You cannot create XML configuration files in Composer as in DAB, but you can import an existing XML application and modify the associated XML application configuration file by using the Composer XML editor:

1. Import the XML application from the repository.
2. Edit the XML application configuration file using the XML Configuration Editor.
3. Install the Composer project.

Creating a Workflow

You can create workflows with the Workflow Manager tool, which is packaged with Composer. The following procedure describes how you would go about creating a workflow and importing it into Composer.

1. Launch Workflow Manager with the launch_wfm.bat file that is located in the <Composer_root>\WorkflowManager directory.
2. Create a workflow template and save it.
3. Launch Composer.
4. Create a new project or open an existing project.
5. Import the Workflow template artifact into the Composer project. The workflow can now be installed with Composer.
6. Install the Composer project or export to a DAR file for installation.

Understanding Documentum supplied reference projects

In Composer, projects can reference another project's resources. This allows many projects to reference one project that contains a useful set of resources. Documentum supplies special reference projects that allow you to use or extend Documentum artifacts, because importing these artifacts in your own project is not allowed. More specifically, the names of Documentum artifacts begin with 'dm' and artifacts that begin with this string are not allowed in user projects.

'Dm' artifacts are still necessary to have in Composer projects, however, and Documentum supplied reference projects provide a safe mechanism to support 'dm' artifacts. Every Composer project that you create automatically references the DocumentumCoreProject, which is a read-only project that provides all of the artifacts that Content Server ships out-of-the-box. The DocumentumCoreProject allows you to extend or use 'dm' artifacts, such as creating a custom subtype of dm_document. If you want to use or extend other 'dm' artifacts that are not contained in DocumentumCoreProject, such as Taskspace types, you must obtain the correct reference project from the EMC download site. For more information, see the *Reference projects* section in the *Composer User Guide.*

Chapter 3

Packaging and installing applications

Composer has introduced new packaging and installation schemes that you should be aware of. Composer projects and DAR files are structured differently from DocApps and DocApp archives. The major differences are listed below:

- Each artifact in Composer is represented with its own XML file. This allows you to better manage your projects in source control because the granularity of Composer projects are at the artifact level.
- The notion of checkin and checkout of objects in DAB is no longer used. Composer projects are developed and built offline and must be installed at the project level.
- Composer projects are installed within Composer and DAR files are installed with headless Composer or the DAR Installer plugin.
- If you have referenced other projects in your project, you need to install them (except for Documentum supplied reference projects) before installing your project.

Application installation with the DAR Installer and Headless Composer

The DAR Installer is recommended for simplicity if you want to install DAR files in an ad-hoc manner. It is a simple user interface that you can use to install DAR files one at a time. It works in a similar way to DAI. For more information on obtaining and using the DAR installer plugin, see *Installing a DAR file with the DAR Installer plugin* in the *Composer User Guide*.

Headless Composer is a non-graphical user interface version of Composer that is used for automated deployments of Documentum applications with Ant scripts. Headless Composer is not intended for development and should only be used for deployment. Headless Composer is recommended in situations where an automated and repeatable process is required. For more information using headless Composer with Ant scripts, see *Managing Projects and DAR Files Using Ant Tasks and Headless Composer* in the *Composer User Guide*.

Installation parameters

In Composer, installation parameters were introduced to specify installation-time values. Installation parameters specify values that are needed at installation time. You can think of an installation

parameter as a variable that must be given a value at installation time. The following procedure describes how to use installation parameters in Composer:

1. Create an installation parameter in Composer. You can specify a default value for the parameter if no value is specified at installation time.
2. Install the Composer project.
3. When prompted, specify values for installation parameters or specify a file that contains the parameter values. If no values are specified, the default values are used.

Pre- and post-installation scripts

In Composer, you can still specify Docbasic procedures when installing Composer projects to a repository. The following procedure describes how to specify pre- and post-installation scripts:

1. Right-click the project and select **Properties**.
2. Select **Install Procedures**.
3. Click the **Select** button for pre or post-installation procedures.
4. Select a procedure from the list or choose to create a new one.

Chapter 4

Source control managing with Composer

DAB and Composer handle versioning of Documentum applications differently. Checkins are done at the DocApp level with DAB, but Composer projects are checked in at the artifact level. When checking in files to source control, you should check in the following items that are located in your Composer project's root folder:

- Artifacts folder
- content folder
- dar folder
- Installation Parameter Files folder
- src folder
- Web Services folder
- All files that begin with a dot in the project folder

The following steps describe at a high level how developers would interact with a Documentum application in a source control system.

1. Developer 1 creates a Composer project and creates artifacts.
2. Developer 1 checks in the Composer project to source control. Each artifact is checked in individually.
3. Developer 2 retrieves the entire Composer project from source control.
4. Developer 2 checks out certain artifacts, modifies, tests, and checks them back into their source control system.
5. If multiple developers need to modify the same Composer project, each artifact is versioned as opposed to the entire Composer project.