Oracle® Documaker

Connector

Installation Guide

12.7.1

Part number: F76382-01

January 2023

Contents

Preface

This document contains information necessary for the installation of Oracle Documaker Connector.

Audience

This document is intended for users who want to install Documaker Connector. Experience installing Oracle Documaker and experience as a system administrator is necessary.

In addition to this guide, implementation of Documaker Connector with the Documaker document source requires familiarity with Oracle Documaker configuration and Document Automation Language (DAL) scripting.

Familiarity with configuration of the content management system you choose is also required. This guide is not a substitute for understanding those products and their documentation. You must have the necessary skills in both Oracle Documaker and your content management system to configure those products to work with the Documaker Connector.

Once familiar with the material in this guide and other prerequisite background information, an administrator should be able to plan and execute an implementation of Documaker Connector and manage day-to-day operation.

Documentation Accessibility

Accessibility of Links to External websites in Documentation

This documentation may contain links to websites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these websites.

CUSTOMER SUPPORT

If you have any questions about the installation or use of our products, please call

+1.800.223.1711 or visit the My Oracle Support website: http://www.oracle.com/us/support/index.html. Go to My Oracle Support to find answers in the Oracle Support Knowledge Base, submit, update or review your Service Requests, engage the My Oracle Support Community, download software updates, and tap into Oracle proactive support tools and best practices.

Hearing impaired customers in the U.S. who need to speak with an Oracle Support representative may use a telecommunications relay service (TRS); information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html. International hearing impaired customers should use the TRS at 1.605.224.1837.

CONTACT

USA: +1.800.223.1711

Canada: 1.800.668.8921 or +1.905.890.6690

Latin America: 877.767.2253

For other regions including Latin America, Europe, Middle East, Africa, and Asia Pacific regions: Visit- http://www.oracle.com/us/support/contact/index.html.

Follow us

https://blogs.oracle.com/insurance

https://www.facebook.com/oraclefs

https://twitter.com/oraclefs

https://www.linkedin.com/groups?gid=2271161

Related Documents

For more information, refer to the following Oracle resources:

- The Oracle Documaker documentation set, specifically:
    - Documaker Enterprise Administration Guide
    - Documaker Connector Developer's Guide
    - Documaker Installation Guide
    - Documaker Administration Guide
    - DAL Reference
- The Oracle WebCenter Content Core Capabilities, previously known as the Oracle Universal Content Management (UCM), documentation set, specifically:
    - Oracle Universal Content Management Product Overview
    - Getting Started with Content Server
    - Oracle Universal Content Management Content Server Installation Guide for (*your specific platform*)
    - Managing Repository Content

Conventions

The following text conventions are used in this document:

Convention	Description
bold	Indicates information you enter.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands, URLs, code in examples, and text that appears on the screen.

Note The screen shots and interface examples presented in this document may differ slightly from the actual software you receive. Late-breaking software modifications and interim patches may change the look of the interface, but should not detract from your ability to use the software with this document.

Chapter 1

Introduction

Oracle Documaker Connector works with Oracle Documaker to archive Documaker output documents into content management repositories, such as Oracle WebCenter Content Core Capabilities, previously known as Oracle Universal Document Manager (UCM).

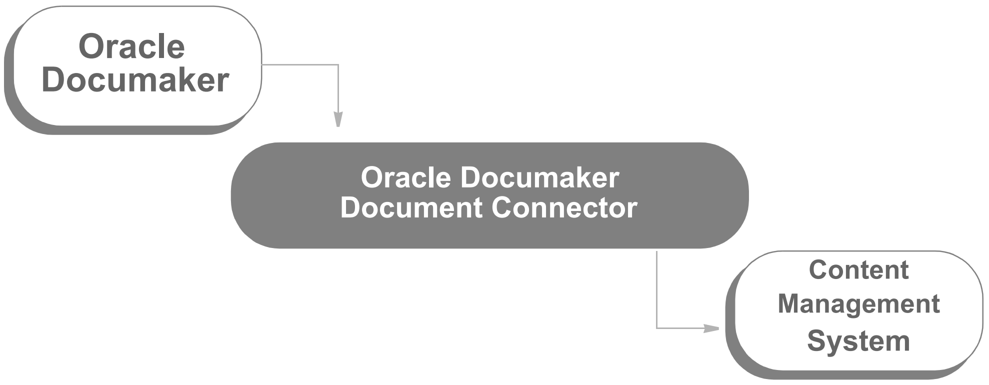
Documaker Connector is also the technology underlying the Archiver function in Documaker Enterprise Edition Document Factory. It is an extremely flexible and adaptable utility. You can find out more about how to customize Documaker Connector in the Documaker Connector Developer Guide.

This chapter provides an introduction to Documaker Connector and covers these topics:

- *Overview* on page 9
- *Compatibility* on page 11
- *Planning* on page 12

OVERVIEW

Documaker Connector works with Oracle Documaker or other sources of documents to archive output documents into content management repositories:


Oracle Documaker
Oracle Documaker Document Connector
Content Management System

Creating document sets with Oracle Documaker

Oracle Documaker is a document publishing solution which is driven by input transactions to produce output document sets. Documaker combines input transaction data and a repository of forms with a powerful rules engine and composes custom documents based on the incoming transaction data.

Each transaction is a set of variable data field values and other form-selection criteria. Oracle Documaker applies the variable data to choose forms, create presentation graphics such as pie charts or bar graphs, and fill in fields on the forms, optionally performing formatting and/or calculations on the data before using it as text on the forms.

Because the output documents were often originally designed to be printed, the output documents and the files containing them are sometimes called *print streams*.

The print streams are not only printed and delivered to the recipients, but are also sent via various forms of electronic delivery such as email or downloaded via the Internet. Once published, the documents also usually need to be retained in that form for reference by customer service or other agents. Electronic filing in a content management system is the job of the Documaker Connector, or in the Documaker Enterprise Edition Document Factory the job of the Archiver component, using Documaker Connector destinations.

Using Documaker Connector

When used with Documaker Connector and the Documaker source component, Documaker’s DAL (Document Automation Language) scripting is used to place each output document in a file system directory and insert a matching record containing the variable data and indexing information, also called *metadata*, into a database table.

Documaker Connector monitors the database table for new records and then processes those records by sending the document and indexing data into a destination document management system. You can run Documaker Connector continuously, as a system service, or periodically, such as after a batch run of Documaker.

You configure Documaker Connector with the connection information for the source database table and the connection information for the document management system. The columns of the database table and the corresponding configuration of the content management metadata vary with the implementation. To configure these items, you need knowledge of both Documaker and the content management system administration. For instance, you must...

- Set up the database table columns to hold the metadata you want to use in your document management system so it can be transferred by Documaker Connector.
- Set up Documaker to fill in those columns for each document record it will insert in the table, using Documaker DAL scripts.
- Create metadata fields in your content management system to receive the data from Documaker Connector.

There are also performance and volume-related configuration parameters.

Setting up Documaker Connector

Documaker Connector is highly customizable and comes with content management destinations for both Oracle's WebCenter Content, Core Capabilities (formerly Oracle Universal Content Management) and WebCenter Content, Imaging (formerly Oracle Imaging and Process Management). It also comes with several testing and general utility connectors, described later.

You can also create destination components for other content management systems so they work with Documaker Connector. It can also be used as a framework for creating new applications which draw documents from sources other than Documaker — you can customize both sides of Documaker Connector.

Note For more information about writing custom components for Documaker Connector, both source and destination interfaces, see the Documaker Connector Developer's Guide

Although you can customize the source and destination sides of Documaker Connector, this guide is primarily concerned with the Documaker source, which reads metadata records from a database table and document contents from files in a directly accessible directory.

Index data (metadata) is read from the database table and referenced by the column names in the table. The metadata is then available to the configured destination as a name-value pair of data, named after the database table column and with the value of that column in the row for the document being processed. It is this set of name-value pairs that Documaker Connector processes and passes to the content management system to use to index the document.

COMPATIBILITY

Documaker Connector is tested with the following:

Application	Version
Oracle Documaker	Version 12.6.2 and is compatible with Documaker versions 11.2 and higher
Oracle WebCenter Content, Imaging	WebCenter Content release 11g, version 11.1.1.9.0

There are additional compatibility requirements for these applications. Consult the product documentation for those products, and in particular the specific versions you plan to run as a part of your system planning.

Planning

Before implementing the Oracle Documaker Connector, there are a few planning steps that need to be completed. You need these answers to configure Documaker, WebCenter Content, and Documaker Connector.

Thinking through and making these decisions up front will make the implementation process go much more smoothly. You will want to involve all stakeholders in the implementation, particularly those who will access the documents from the content management system.

You should consider several questions and make sure everyone is in agreement and understands the implications of the decisions. This implementation and any subsequent changes will flow through the entire process. Resolve these questions before you implement the system:

- Which documents will be archived?
- Which output format, such as PDF, will be sent to the content management system?
- What variable fields will be used to organize and retrieve the documents? These fields contain the *metadata* used to index the documents into the content management system.

Define the Content

First decide which of your documents you want to save in the archive. You may be publishing several versions of the same basic document, such as different recipient copies.

There may not be any reason to archive multiple versions of the same document. On the other hand, if you have multiple uses for the archived documents, possibly with different end users of the documents, you may find it convenient.

For example, you may want to archive a master copy for customer service to reference with both customer and internal forms included, but you may also want to serve documents to a customer web portal where they can retrieve only the forms visible to them in their final document. Another example might include a utility billing application which serves individual customers copies of their bill with additional marketing or usage analysis pages, but also archives a ledger copy that includes only the current period financial data for use in customer service or accounting.

Define the Format

Next, for those documents, you must decide what output format you want to send to the content management system. This should be a format which your end users can retrieve and view easily. It must also be a format which supports all anticipated uses of the documents you are archiving.

A popular choice is Adobe's Portable Document Format (PDF). It is a cross-industry standard format that provides high fidelity output, cross-platform support, and is universally viewable. Documaker can produce these types of PDF files:

- Text PDF files, where the text in the documents is searchable and can be set to allow copying to other documents.
- Image PDF files, which can be generated by print to disk processes or document conversion using a printer metaphor. These PDF documents contain images or pictures of the pages, similar to scanned documents. They are generally much larger and less useful than text PDF documents.

Determine How You Will Retrieve the Documents

Finally, you must define the information users will use to search for the documents they want. Remember, this is data your content management system users will have on hand to use to search for the documents they want to retrieve. This is generally a subset of the variable data published on the documents. Here are some examples:

- Customer/account number or ID
- Customer name
- Date of issue
- Telephone number
- Postal code

This data is called *metadata*, meaning data about your data — in this case, data about your documents. It must be saved into a database table which you customize, so it can be read directly from there by the Documaker Source. Writing these database records is done by a Document Automation Language (DAL) script, which you create. Each row in the table represents a document to be archived and contains the metadata by which it will be indexed in the content management system.

To design the target layout for your metadata, you need to understand the options available in your content management system for placement of this data. This also helps you understand how to configure Documaker Connector for your destination content management system, since the options available in each system influence the way the destination is configured and how the data coming out of Documaker is mapped.

For example, some content management systems keep all the metadata together with the documents themselves and store documents in a rather flat structure. Other systems let you create one or more levels of folders to collect and organize documents for the users. These folders can have metadata attached to them (sometimes called *properties*) in some systems which may or may not be the same as the document metadata. Depending on the destination system, you can map your metadata to document or folder attributes, or some other container or object in your content management system.

Using Oracle WebCenter Content

Basic metadata fields are preconfigured in the WebCenter Content server database and you cannot edit them. The basic fields built into the system include:

Content ID	Title	Author
Type	Security Group	Revision
ID	Check In Date	Indexed Date
Release Date	Expiration Date	Checked out
Checked out by	Revision Status	Indexer Status
Conversion Status	Indexer Cycle	Workflow State
Revision Rank	Publish Type	Publish Status

Folders are an optional feature of WebCenter Content. When the feature is installed/ enabled folder objects have the same set of basic metadata and additional fields are added to both documents and folders. For example, a Folder field is added to each document to hold the parent folder ID if the document is in a folder.

Custom metadata is added in a single screen in the Content Server's Configuration Manager application. They are called *information fields*. When fields are added to this list, they become attributes of all documents and folders.

Chapter 2

Installing Documaker Connector

The installation of Documaker Connector places the necessary program files and supporting software on the target system at the selected location. This chapter guides you through installing the software and prepares you for configuring the system to meet your requirements.

This chapter discusses these topics:

Note Depending on the database system you chose and the content management system, you may require underlying supporting software which is not provided as part of the Documaker Connector installation. See the documentation about your chosen document source and destination interfaces for specific requirements.

SOFTWARE REQUIREMENTS

To use Documaker Connector, you need the following:

- Platform support of Java 8 (compatible JVM), including JDBC database connectivity.

Note On Windows and Linux the installer installs a compatible JVM (Java Virtual Machine) for you.

- Connection to a SQL database and file system directory which are both also accessible to Oracle Documaker, for use with the Documaker source.
- A connection to the configured destination system, such as Oracle WebCenter Content.

Note For detailed information see Documaker System Requirements.

DOWNLOADING THE SOFTWARE

The Oracle Software Delivery Cloud (OSDC) site lets you download Oracle software products.

The process of downloading software from OSDC includes following steps:

1. First, go to Oracle Software Delivery Cloud website. http://edelivery.oracle.com/
2. Sign in with your Oracle account. If you do not have an Oracle account, you can register for an account here.
3. Search for the software by typing in the search bar and selecting it. For example enter 12.6.2 to search for the list of Release 12.6.2 versions of Oracle Documaker software products.
4. Select the platform from the 'Select platform' drop-down.
5. The selected products are then listed under 'Download Queue'. Click the X (cross) which is adjacent to the product in case you want to remove individual files or click 'Remove All' in the lower left corner of the dialog if you want to remove all the listed items.
6. Click 'Continue' to proceed to next screen; you will see a list of the selected software for downloading.
7. Choose the individual software components for download and click 'Continue' if you wish to proceed or 'Return to Search' to review different software for downloading.
8. Read the license agreement carefully; mark the check box to agree with license agreements, and click 'Continue'.
9. Click 'Download' button to download the software or click the filename to individually download the files.
10. While you can save the file on any machine you choose, we recommend you save the file onto the machine where you plan to run it. You must unzip the file on the platform for which it was intended. The length of time it takes to download an application depends on the size of the download, your connection speed, and the amount of traffic on the site.
11. Once the Download has completed, click 'Return to Search' to search and download additional files or click 'Sign Out' to log off Oracle Software Delivery Cloud.

INSTALLING DOCUMAKER CONNECTOR

Documaker Connector is installed using platform-specific setup wizards which place the program and supporting files on the target system and perform the initial configuration of the target system to run Documaker Connector.

For instance, on Microsoft Windows systems, this includes installation of Documaker Connector as a Windows Service and installation of a task bar notification icon which you can use to control the running of the service.

To install on	Required user privileges	See
Windows	You must log in as the Administrator to install the Windows Service. You must also have write-permission to the disk directory where Documaker Connector will be installed. A non-Administrator can run the Setup wizard, however, the installation of the Windows service will fail. The program files will be installed and you can run the program as a command-line program, but it will not be added to the Services list in the Services control panel.	on page 18
UNIX (in GUI mode)	You must have write-access to the target directory. No special privileges are required.	on page 18
UNIX (via the command line)	You must have write-access to the target directory. No special privileges are required.	*Installing from the UNIX Command Line* on page 22

USING THE SETUP WIZARD

Running the Setup wizard is similar across all platforms. These steps are from Microsoft Windows, but should be easily used as a guideline in other environments.

Note On UNIX systems (Linux), to optionally run the installation in GUI mode you must have an X11 server running. You must also set the DISPLAY environment variable to point to the X11 server, otherwise it will run in a console mode. For more information, see *Installing from the UNIX Command Line* on page 22.

Follow these steps to run the X11 Java Swing-based GUI:

1. Start the X11 server on your computer or use the existing X11 hosted Linux desktop environment.
2. Set the DISPLAY environment variable to point to the X11 server. Here is an example:

```
DISPLAY=IPADDRESSOFX11Server:DisplayNumber
```

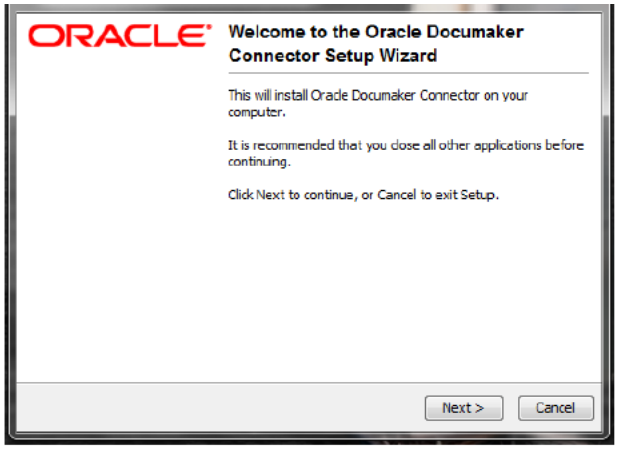
3. Run the Setup wizard.

From the directory into which you downloaded the Oracle Documaker Connector media pack, unzip the media pack, locate and double click on the following program:

```
documakerconnectorrel(version and patch number).exe
```

The installation wizard starts. Follow these steps to install Documaker Connector:

1. When the Welcome window appears, click Next.

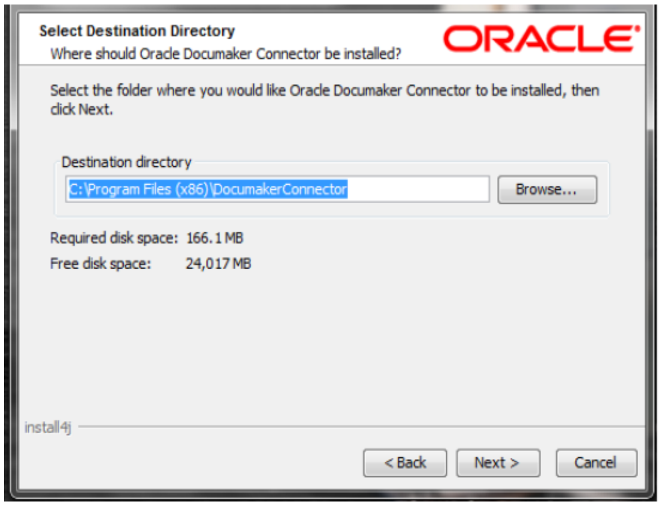

Welcome to the Oracle Documaker Connector Setup Wizard

This will install Oracle Documaker Connector on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

The Select Destination Directory window lets you choose the installation target directory.



Select Destination Directory
Where should Oracle Documaker Connector be installed?

Select the folder where you would like Oracle Documaker Connector to be installed, then click Next.

Destination directory
C:\Program Files (x86)\DocumakerConnector

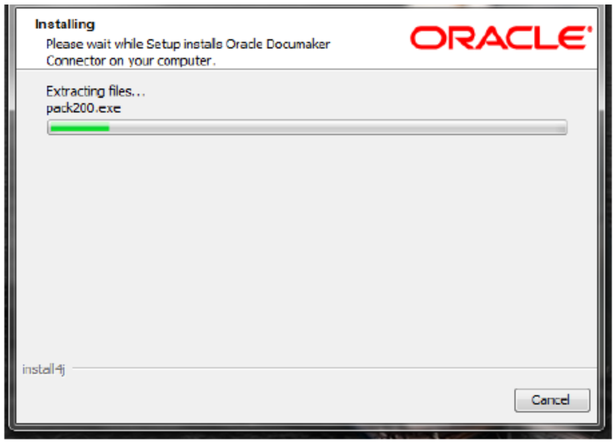
Required disk space: 166.1 MB
Free disk space: 24,017 MB

2. Enter a directory or click Next to accept the default directory. On Windows, the default directory is under the Program Files directory. The directory you choose is referred to throughout this document as the installation target directory.

   Once you click Next, the Setup wizard begins to install Documaker Connector. As part of this process, the Setup wizard...

   - Puts a Java run-time environment (JRE) in a subdirectory of the target directory called *jre*. On Microsoft Windows, the Setup wizard installs the Visual C++ 2008 run time if it is not already on the machine.
   - Sets the service.path in the dm_connector.svc.properties file to the JRE that was installed.
   - (On Windows) Runs the dm_connector_svc.exe program with an *install* parameter to install the Windows Service. Running this program later with a parameter of *uninstall* removes the Windows Service.

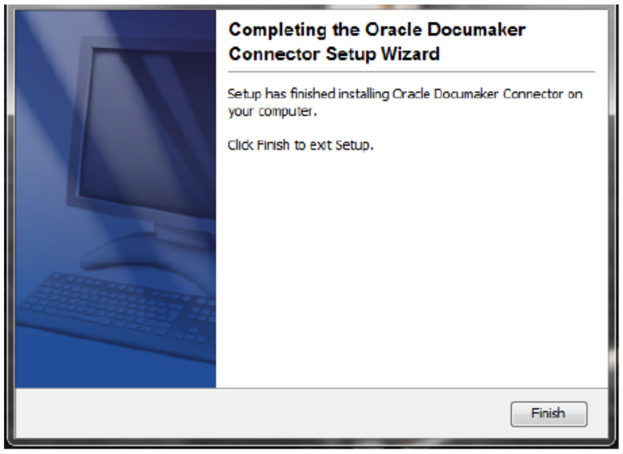
   This window shows its progress:



Installing
Please wait while Setup installs Oracle Documaker Connector on your computer.

Extracting files...
pack200.exe

When finished, this window appears:



Completing the Oracle Documaker Connector Setup Wizard

Setup has finished installing Oracle Documaker Connector on your computer.

Click Finish to exit Setup.

3. Click Finish to complete the installation.

Installing from the UNIX Command Line

Running the installer from the UNIX command line avoids the requirement of having a windowing or GUI environment. All interaction with the installer is handled via the text console. The installer is packaged as a shell script. Here is an example:

```
DocumakerConnectorRel121p01b15380Linuxx86.sh
```

Note This example is for Linux on x86, but other UNIX platforms are very similar. The name of the file you download for your platform or software version may differ from the exact name shown in the example.

Follow these steps to install the software from a UNIX command line:

1. Once the script is copied into your UNIX directory, it must be made executable if it is not already. Use the chmod command to set the appropriate read and executable attributes. Here is an example:

   ```
   ~> chmod 555 DocumakerConnectorRel121p01b15380Linuxx86.sh
   ```

2. After you set the read and executable mode of the file, run the script:

   ```
   ~> ./DocumakerConnectorRel121p01b15380Linuxx86.sh
   ```

   The script takes a few seconds to start and begin unpacking the compressed files contained within it. Here is an example of the text that appears on your console:

   ```
   ~> ./DocumakerConnectorRel121p01b15380Linuxx86.sh
   Unpacking JRE ...
   Preparing JRE ...
   Starting Installer ...
   This will install Documaker Connector on your computer.
   OK [o, Enter], Cancel [c]
   ```

3. Press Enter to continue with the installation. Here is an example of the text that appears on your console:

   ```
   Where should Documaker Connector be installed?
   [/home/example/DocumakerConnector]
   ```

4. Enter a different location if necessary or press Enter to accept default location and continue with the installation. Here is an example of the text that appears on your console:

   ```
   Preparing to copy files...
   Extracting files...
   Downloading ...
   Extracting files...
      COPYRIGHT
      README
      THIRDPARTYLICENSEREADME.txt
      Welcome.html
      bin/
      bin/tnameserv
      ... (the complete list of installed files is omitted here)
      lib/oracle-ridc-client-11g.jar
      .install4j/
      .install4j/uninstall.png
      uninstall
   Finishing installation...
   ~> cd DocumakerConnector
   ~> ls
   total 41
   ```

```
   -rw-r--r--  1 example users  2930 2011-03-08 21:37 batch-file-
conn.properties
   -rw-r--r--  1 example users  1330 2011-03-17 12:55
dm_connector_svc.properties
   -rw-r--r--  1 example users  3746 2011-03-08 21:37 dmkr-ucm-
conn.properties
   drwxr-xr-x  4 example users   232 2011-03-17 12:55 jre
drwxr-xr-x  2 example users  1128 2011-03-17 12:55 lib
   -rw-r--r--  1 example users  2888 2011-03-08 21:37 mock-file-
conn.properties
   -rw-r--r--  1 example users  3042 2011-03-08 21:37 mock-ftp-
conn.properties
   -rw-r--r--  1 example users  2758 2011-03-08 21:37 mock-mock-
conn.properties
   -rw-r--r--  1 example users  3154 2011-03-08 21:37 mock-ucm-
conn.properties
   -rwx------  1 example users 10337 2011-03-09 18:53 uninstall
```

Note During the installation on UNIX, you may see a warning message regarding the Java system preferences store at /etc/.java/.systemPrefs/com. This warning should not affect your installation.

RUNNING THE INSTALLER WITH A RESPONSE FILE

You can run the installer in unattended mode from the command line. To do this you must create a response file to provide the information the installer needs. A response file is a text file which contains the information a user would typically provide while running the Setup wizard or responding to command-line prompts. This information is in this format:

```
name=value
```

To create a response file that contains the necessary data, first run the command-line installer in one of these ways:

- Using default mode, with no arguments. After the installation finishes, the installer creates a response file named *response.varfile*.
- Using the command-line installation with the *-console* parameter. This lets you specify the name you want to assign to the response file.

After the installation finishes, the installer creates the response file and stores it in the .install4j directory. This file contains name=value data captured during the installation process. You can edit this file if necessary to modify the values. Here is an example:

```
#install4j response file for (application/version)
#Wed Mar 16 16:53:12 EDT 2011
sys.languageId=en
sys.installationDir=d\:\\(application/version)
```

You can then pass the response file to the installer using the *-varfile* parameter. For example, to run the installer in unattended mode using a response file, include these parameters:

```
-q -console -c -varfile varfilename
```

Parameter	Description
-q	Runs the installer in unattended mode.
-console	If the installer is executed in unattended installation mode (-q) and you include -console as the second parameter, a console is allocated on Windows that displays the output of the installer.
-c	Runs the installer in the console mode.
-varfile	V*arfilename* specifies the name of the response.varfile to use. You can include a full path.

Note You can use a response file on both Windows and UNIX/Linux installations. For more information about response files and install4j installations, go to this website:

http://resources.ej-technologies.com/install4j/help/doc/

Here is an example:

```
(application/version).exe -q -console -c -varfile
  response.varfile
```

Here is an example of the output you will see:

```
Extracting files...
Downloading ...
Extracting files...
Finishing installation...
```

Checking Your Installation

Although there are slight differences by platform, most of the installed files are the same regardless of the target system. The following example is from a Windows-based installation.

Setup wizard files

The Setup wizard places its own support files in the .install4j subdirectory. This directory should not be disturbed, as it contains information used if you decide to remove Documaker Connector. For more information, see *Removing Documaker Connector* on page 32.

Java run-time files

The Setup wizard places a Java 8 run-time environment (JRE) for the correct platform in the jre subfolder of the installation target directory. The configuration files for Documaker Connector are set up to use this JRE. If you need to use another JRE, you can reconfigure the properties and script files to use another JRE in the same or different location.

Java application and support files

All the Java JAR files for the application and the required support code are installed in the lib subdirectory. These JAR files must be in the Java classpath for the Documaker Connector to run (either from the command line or as a service). All of these files are installed as part of the normal installation process, but the list is provided here for information useful in troubleshooting or customizing the install.

The default logging properties file log4j.xml is installed in the lib folder, with the .jar file listed in this table. Some of the files are 3rd-party libraries which are available via the Internet. For these files, the URL you can use to download the file is provided, in case you need it.

Name	URL
commons-beanutils.jar	http://commons.apache.org/beanutils/
commons-codec-1.6.jar	http://commons.apache.org/codec/
commons-collections-3.2.jar	http://commons.apache.org/collections/
commons-dbcp-1.2.2.jar	http://commons.apache.org/dbcp/
commons-httpclient-3.1.jar	http://hc.apache.org/httpclient-3.x/
commons-lang-2.3.jar	http://commons.apache.org/lang/
commons-logging-1.1.1.jar	http://commons.apache.org/logging/
commons-net-2.2.jar	http://commons.apache.org/net/
commons-pool-1.3.jar	http://commons.apache.org/pool/
DdlUtils-1.0.jar	http://db.apache.org/ddlutils/

Name	URL
jakarta-oro-2.0.8.jar	http://jakarta.apache.org/oro/
log4j-1.2.15.jar	http://logging.apache.org/log4j/1.2/
ojdbc14.jar	http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-10201-088211.html or http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html
oracle-dmkr-conn-assuresigndestination.jar	Oracle proprietary file
oracle-dmkr-conn-batchloadersource.jar	Oracle proprietary file
oracle-dmkr-conn-documakersource.jar	Oracle proprietary file
oracle-dmkr-conn-documentdatasource.jar	Oracle proprietary file
oracle-dmkr-connector.jar	Oracle proprietary file
oracle-dmkr-connectorapi.jar	Oracle proprietary file
oracle-dmkr-conn-filedestination.jar	Oracle proprietary file
oracle-dmkr-conn-fileplusdestination.jar	Oracle proprietary file
oracle-dmkr-conn-ftpdestination.jar	Oracle proprietary file
oracle-dmkr-conn-ipmdestination.jar	Oracle proprietary file
oracle-dmkr-conn-mockdestination.jar	Oracle proprietary file
oracle-dmkr-conn-mockperiodicprocess.jar	Oracle proprietary file
oracle-dmkr-conn-mockreporter.jar	Oracle proprietary file
oracle-dmkr-conn-mocksource.jar	Oracle proprietary file
oracle-dmkr-conn-pdfburster.jar	Oracle proprietary file
oracle-dmkr-conn-ucmdestination.jar	Oracle proprietary file
oracle-dmkr-ezridc.jar	Oracle proprietary file
oracle-dmkr-scrambler.jar	Oracle proprietary file
oracle.ucm.ridc-11.1.1.jar	Get this file from your UCM installation in the directory: <Middleware Home>\<ECM Home>\ucm\Distribution\RIDC

Table 1: Required Java Library Support Files (JARs)

Note In addition to the files supplied with the application, you must make sure the appropriate JAR files for your database provider are also included the classpath. These files are not included in the installation of the Documaker Connector — with the exception of the Oracle JDBC JAR file.

Here are some examples of the JAR files required for the JDBC connectivity for various databases:

Database	JAR file
Oracle	ojdbc14.jar (included with the application)
IBM DB2	db2jcc.jar
MySQL	mysql-connector-java-5.1.5-bin.jar
Microsoft SQL Server	sqljdbc.jar
Note: Vendors may revise these names with new versions of their products.	

Table 2: JDBC Library Support Files (JARs)

Configuration and script files and Windows utility programs

At the root of the target installation folder, scripts, and configuration property files for Documaker Connector are placed along with utility programs useful on Microsoft Windows. These files are described here:

File	Description
dm_connector_agent.exe (Windows only)	This program creates the Task Tray Agent and provides convenient control of the Documaker Connector Windows Service
dm_connector_svc.exe (Windows only)	This program can be run as a Windows service and configured to run a Java program, like Documaker Connector, as the actual service program.
dm_connector_svc.properties Windows only)	The configuration file for dm_connector_svc.exe which causes it to run Documaker Connector as the real service and sets the CLASSPATH for the Documaker Connector program as well as the Documaker Connector properties file. It includes the path to the JVM and the Documaker Connector mode to be run as a service.
ecmconnector.bat (Windows only)	Batch script which may be useful to run Documaker Connector in a command prompt window. This file is described more later.
uninstall.exe (Windows only)	Lets you uninstall Documaker Connector.
source-dest-conn.properties	Series of example files providing a starting point for configuration of Documaker Connector with specific combinations of a source and a destination. The specific files are: • batch-file-... - Oracle WebCenter Content-batchloader source to a file-system folder destination • batch-ucm-... - Oracle WebCenter Content-batchloader source to Oracle WebCenter Content destination • dmkr-ucm... - Oracle Documaker source to Oracle WebCenter Content destination • mock-file... - Test data source to file-system folder destination • mock-ftp... - Test data source to FTP-server destination • mock-mock... - Test data source to Test data (null) destination • mock-ucm... - Test data source to Oracle WebCenter Content destination

File	Description
ecmconnector.sh	A UNIX shell script useful to run Documaker Connector.
vcredist_x86.exe (Windows only)	Microsoft installer for run-time support for the above applications. Provided in case the target install computer does not have the required support files.

Table 3: Configuration and script files and Windows utility programs

Setting Up Documaker Connector as a Windows Service

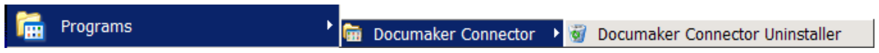
The dm_connector_svc.exe application runs Documaker Connector as a Windows Service. For more information, see *Windows Service Application* on page 79.

The dm_connector_agent.exe program places the Task Tray Agent icon in the Windows task bar:

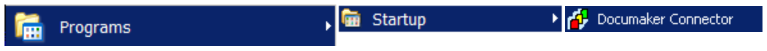
This icon shows the status of the Documaker Connector Windows service. Right-click on the icon to open the log file and start and stop the Windows service without opening the Services Control Panel.

USING THE TASK TRAY CONTROLLER

When the Setup wizard finishes, you will find a Documaker Connector Start menu folder with an uninstall option:

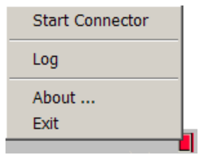


It also places the Documaker Connector Task Tray Agent in your Startup folder:



Select this option to start Documaker Connector Task Tray Agent. The Task Tray Agent appears in your task bar. Here is an example:

You can start the Task Tray Agent and then use it to start the service. Right click on the Task Tray Agent and choose the Start Connector option to start Documaker Connector:



The icon is yellow while starting up and then turns green while running normally.

Note The default configuration of the service is to run Documaker Connector with the Documaker Source and the WebCenter Content destination. Before running the service, you must complete the configuration so the Documaker Source can connect to the database and the WebCenter Content destination can reach the WebCenter Content system. For more information, see *Configuring Documaker Connector* on page 33.

Downloading Patches

You can download the latest Oracle software patches at the My Oracle Support website. The process includes:

- Going to the My Oracle Support site (requires registration)
- Searching for the patches you want to download
- Downloading those patches

To download Oracle software patches, go to the My Oracle Support website:

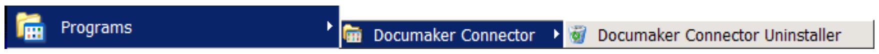
https://support.oracle.com

Removing Documaker Connector

Before removing Documaker Connector on a Windows system, be sure the application is not running as a Windows Service. To do so, first check the task tray controller or open the Services administrative tool (select Control Panel, Administrative Tools, Services) and stop the service if it is running. Then run the service program with the *uninstall* parameter to remove the registration of Documaker Connector as a Windows Service:

```
dm_connector_svc uninstall
```

To uninstall Documaker Connector, choose the Documaker Connector Uninstaller option form the Start menu:



On UNIX, the installer places an *uninstall* shell script in the installation target directory. To uninstall Documaker Connector, run the script from the parent directory of the installation target:

```
./DocumakerConnector/uninstall
```

Chapter 3

Configuring Documaker Connector

Once installed, you must configure Documaker Connector to work with your Oracle Documaker Standard Edition (ODSE) implementation and to connect and import into the content management system.

The topics in this chapter describe these steps, starting with Documaker where the document sets are created, then on to the content management system where these document sets are stored.

This chapter provides information on the following topics:

Note Installations of Documaker Enterprise Edition (ODEE) are configured differently. The Archiver component of Document Factory replaces the Documaker Connector. You simply configure the proper Archiver destination to your content management system.

OVERVIEW

Setting up Documaker Connector to move document sets created by Documaker into a content management system such as WebCenter Content involves these tasks:

- *Configuring Documaker* on page 37
- *Configuring Oracle WebCenter Content* on page 44
- *Configuring Documaker Connector* on page 47

Note You can do these steps in any order, but you must complete all of them before you run Documaker Connector.

If you customize Documaker Connector to work with a source other than Documaker, the same overall planning and strategy described here may still be applied to that source and destination. Many of the considerations will be the same, such as

- What type of documents to produce
- What metadata is available
- How to organize it in the destination system
- How to optimize the performance of Documaker Connector processing

Both your source and destination systems will require proper configuration in addition to the configuration of the Documaker Connector components to coordinate the transfer of your metadata from the source to the destination, along with the documents themselves.

Archiving documents with Documaker Connector is a cooperative process involving Documaker, Documaker Connector, and your content management system. All must be properly configured for the process to work reliably. The most critical aspect of this configuration is the set of indexing metadata with which the documents are to be archived. As described in *Planning* on page 12, you should already have decided which variable fields will be used as metadata. If you have not completed that step, stop now, and get that list together.

The indexing metadata must be supplied for each document by the Documaker process (generally as part of the incoming extract variable data for each document). Documaker must be configured to pass the desired data to Documaker Connector so it can, in turn, use it to archive each document. Configuring each step in the process so that they all agree on the indexing data from end to end requires care and planning.

An overview of the process follows.

Setting Up Documaker

Your content management system probably requires configuration to create fields to store the indexing data, or *metadata*. This, too, can be done before installing Documaker Connector.

You must set up Documaker to produce the documents to be archived along with the desired indexing data. You can do this step before or after you install Documaker Connector. This step requires expertise in the following areas:

- Configuring Documaker
- Using Document Automation Language (DAL) scripting
- Using SQL database table definitions

Documaker Connector with the Documaker source is driven by records it retrieves from a database table. Each record represents a document copy produced by a Documaker batch process.

If you are using Documaker Enterprise Edition with Document Factory, these database records are part of the Document Factory processing. Documaker Standard Edition must write the records itself.

The database records contain some fields used to manage the import process, but most importantly contain the data to be imported into your content management system as indexing data. Documaker must write this data into the database record to pass it along to the content management system. You must configure Documaker to properly write these database records along with the documents.

These document copies are intended to be imported into your content management system for the purpose of providing an archive of record. Oracle Documaker is usually configured using Batch Banner and Transaction Banner DAL scripting to control the output of the documents and to write the database records into the proper table.

Be sure to enable both Batch and Transaction Banner processing at the individual Batch:XXX control group being used to write the documents files as well as at the overall Printer control group in Documaker. For more information, see *Example INI Configuration* on page 40.

Once this step is completed, check the results by running Documaker and manually validating the contents of your database table. Adjust your configuration until the records are properly written into the database table. You do not have to have Documaker Connector installed to complete this setup.

Setting Up the Content Management System

To produce these documents, Documaker must have a *recipient* configured that will generate the version of the documents you want to archive. This can be a recipient who is actually receiving the documents or a recipient you created for the purpose of creating archive-versions of the documents.

You must configure a Documaker printer definition to produce the format you want to archive, such as PDF, and write it to a file in the appropriate directory. A batch is then set up for the recipient with the output printer definition. In this batch, you also set up four banner scripts which implement the metadata output to the database and allow for creation of file system sub-directories, and so on. This is described in the topics that follow.

You must configure your content management system with document or folder (or other) attribute fields appropriate to hold the desired indexing metadata. Your users will find the documents you archive by searching for them using these metadata fields. For example, a call center might need to search for documents based on a customer number, telephone number, or customer name. These fields need to be in the database fields, in addition to being in the document content.

Your system may be able to hold the data as attributes of the individual documents and/or document containers such as folders or some other object your system defines. In any case, you will have to make a map of the incoming metadata from your document source to the outgoing data into your destination, field by field. Within the destination content management system, use the administrative tools provided by your system to create the fields or attributes to be filled by the incoming data with each document.

Setting Up Documaker Connector

Lastly, you need to configure Documaker Connector so it uses the Documaker source component and so the Documaker source component can...

- Find the database table produced by the Documaker DAL scripting
- Access the documents referenced or stored in the database table

Also, you must configure Documaker Connector to use the destination component you chose and so that destination component can...

- Connect to the destination content management system
- Write the documents to the content management system so those documents can be archived

In addition to these basic connectivity configuration requirements, Documaker Connector has configuration options for logging its operation and adjusting performance characteristics of the importing process which it manages. These settings allow Documaker Connector to be scaled from a small proof of concept to a large scale production environment.

CONFIGURING DOCUMAKER

Note The information in this topic generally explains the required Documaker Server setup. It provides some examples so a knowledgeable Documaker administrator can apply the information to their installation and generate the documents and metadata for use with the Documaker Connector's Documaker source component.

Documaker Connector with the Documaker source reads a database table for rows containing document metadata and an operating system path to each document file. The Documaker source performs these tasks...

- Queries the database table for rows that have a status field that indicates those records have not been processed (STATUSCD column value of 0)
- Selects these rows with a maximum row count which equals as many records as it is configured to consider as a maximum sized batch
- Marks them with a status value (STATUSCD column) of 3 (in progress)
- Processes each of those records

Documaker Server must be configured to properly generate both the document files and the accompanying database records for the documents to be archived. The configuration of Documaker Server aligns with the configuration of your content management system so Documaker will provide the desired metadata/indexing elements.

Once this list of data elements is determined, you can configure Documaker Server to store the data elements in a database table for Documaker Connector. Documaker Server can store extra data in the table that is not used for indexing, but it must at least provide all the metadata elements which are mapped into the destination content management system.

Documaker Server reads the variable data for each document from a data extract input file or database. This file is supplied in either XML or CSV (comma-separated value) format. To use the data elements in Documaker Server, the data extract values are copied into Documaker global variables, also known as GVM variables or GVMs, using one of these methods:

- TRN_FIELDS INI option
- Ext2GVM rule

Once the data are in GVM variables, you use DAL scripting to control a recipient batch's output print stream name and map the metadata in the GVM variables to database table columns. The Batch Banner and Transaction Banner processing DAL scripts insert the database rows.

Batch and transaction banner processing DAL is used since DAL scripts can be triggered for the different phases of output generation. The output print stream file name and location can be controlled as well as the post transaction processing step to map the GVM variables and any other static data to database table columns (using the DBPreVars DAL function) and finally insert database rows (using the DBAdd DAL function).

Making Index Data Available in Documaker GVM Variables

You can map Documaker extract input data, which are in XML format, into GVM variables using the Trn_Fields INI option via XPATH notation. This requires setting up the mapping with XPATH declarations and modifying or creating the associated TRNDFDFL.DFD (transaction) and RCBDFDFL.DFD (recipient batch) Data Formation Definition files.

DFD files are used by the Documaker data storage abstraction interfaces. The data can be ASCII files, database files, and so on. Documaker pulls the data from extract files (usually in XML or ASCII format) in batch processes and pushes them into the field names defined in the DFD file. This storage is then used as input to other batch processes.

Overall, these files control the storage and propagation of the data through the batch system. See the Documaker Administration Guide for configuration details. For more information, see *Example XML Extract File* on page 81.

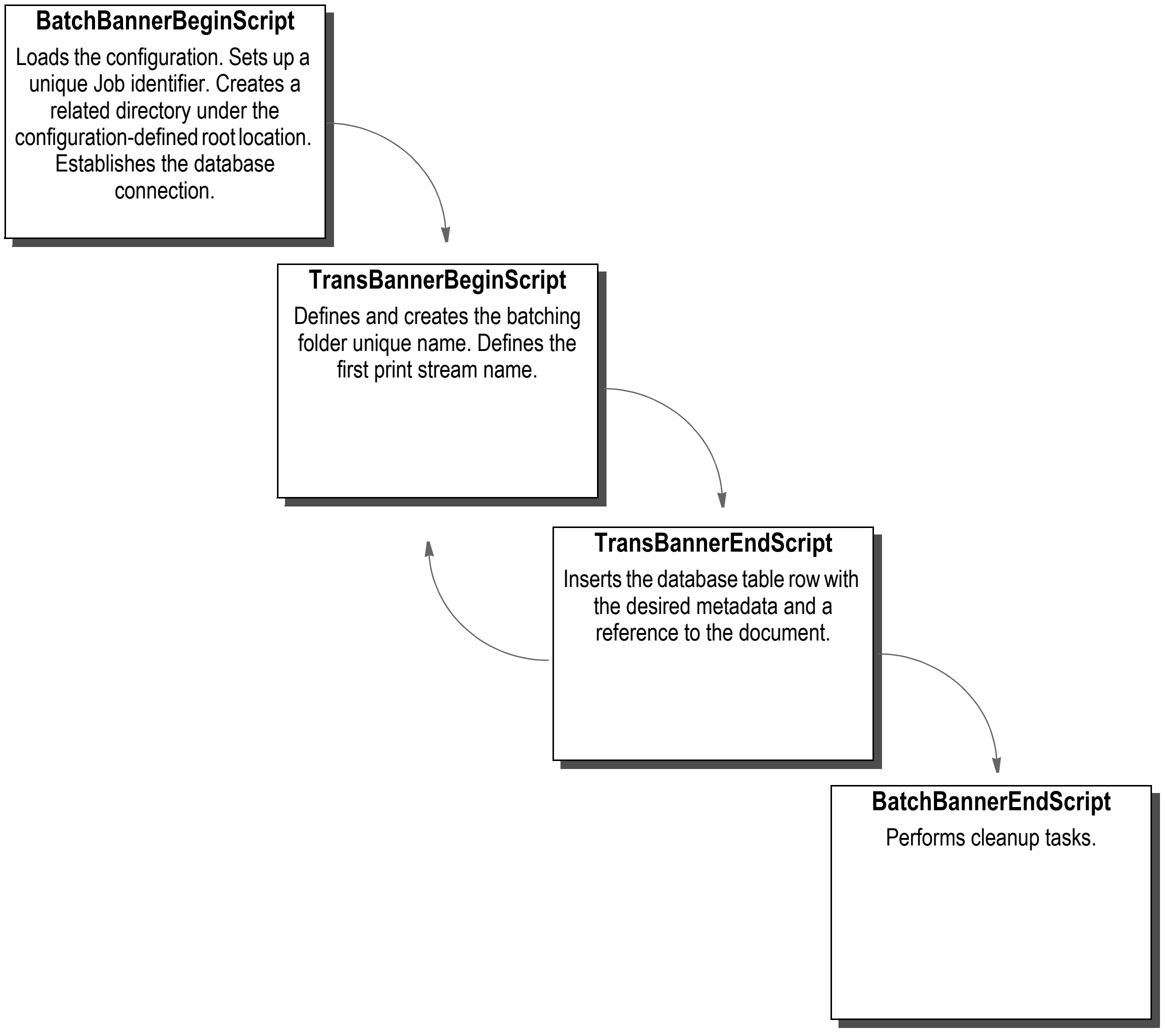
You can also use the Ext2GVM rule to map the data. See the Rules Reference for more information on this rule.

Generating Database Table Rows and Writing the Document Files

You must create a database table and configure Documaker Server to both create the uniquely named recipient batch output print streams and to insert a row in the table with the name and location of the each output file. Each table row also contains the metadata fields needed for content management ingestion. The Batch Banner and Transaction Banner processing DAL scripts are used to write the output documents and to insert the database table rows referencing these documents.

Note You can find examples of these scripts and other configuration files in *Sample Applications and Files* on page 78.

Batch Banner and Transaction Banner DAL processing each have a Begin and End phase with a separate script for each phase. The phases and scripts are run in this sequence:


BatchBannerBeginScript
Loads the configuration. Sets up a unique Job identifier. Creates a related directory under the configuration-defined root location. Establishes the database connection.
TransBannerBeginScript
Defines and creates the batching folder unique name. Defines the first print stream name.
TransBannerEndScript
Inserts the database table row with the desired metadata and a reference to the document.
BatchBannerEndScript
Performs cleanup tasks.

Example INI Configuration

Enable the Batch and Transaction Banner scripts using the FSISYS.INI or FSIUSER.INI files. Typically, you enable the scripts for the chosen recipient batch output.

In the example implementation, Batch6 is the chosen output which is the FILE recipient's batch and is associated with Printer6. Note that the EnableBatchBanner and EnableTransBanner settings are required in both the Printer control group and the Batch6 control group. Here is an example configuration snippet with the changes and additions (in *italics*) to write PDF output for Batch6 and enable the DAL scripting calls:

```
< BatchingByRecip >
   Batch_Recip_Def         = TRUE;"BATCH1";AGENT
   Batch_Recip_Def         = TRUE;"BATCH3";INSURED
   Batch_Recip_Def         = TRUE;"BATCH2";INSURED
   Batch_Recip_Def         = TRUE;"BATCH4";CLAIMANT
   Batch_Recip_Def         = TRUE;"BATCH4";OWNER
   Batch_Recip_Def         = TRUE;"BATCH4";LIENHOLDER
   Batch_Recip_Def         = TRUE;"BATCH6";FILE
   DefaultBatch            = ERROR
< Print_Batches >
   BATCH1                  = data\agent.bch
   BATCH2                  = data\insflat.bch
   BATCH3                  = data\insured.bch
   BATCH4                  = data\other.bch
   BATCH5                  = data\lienholder.bch
   BATCH6                  = data\file.bch
< Printer >
; Must generally enable banner processing for it to work.
   EnableBatchBanner       = Yes
   EnableTransBanner       = Yes
   PrtType                 = PDF
< Printer6 >
   Port                    = data\file.pdf
   PrtType                 = PDF
   AORDebug                = No
   AORExt                  = .pdf
   AORFilesPerBatch        = 1000
   AORPath                 = c:\AOR\
;  Enable a DAL library of scripts to be pre-loaded
< DalLibraries >
   LIB                     = aor
;  Enable the Banner and Transaction DAL Scripting
< BATCH6 >
   EnableBatchBanner       = Yes
   EnableTransBanner       = Yes
   BatchBannerBeginScript  = AOR_PREB
   BatchBannerEndScript    = AOR_POSTB
   TransBannerBeginScript  = AOR_PRET
   TransBannerEndScript    = AOR_POSTT
   Printer                 = Printer6
```

Example DAL Script

As indicated above in the DalLibraries control group, LIB entry, Documaker looks in the AOR.DAL file for the scripts listed in the Batch6 control group. The AOR.DAL file is a library with the definitions of the Batch6 DAL scripts shown in *Using DAL to Output to a Database Table* on page 93.

As you can see in the file, scripts are provided for the Begin and End events at both the Transaction and Batch levels.

This DAL scripting requires Documaker Server to be configured to connect to a database and for the referenced table to exist in that database. The table schema must agree with what the DAL inserts into that table. You do this by performing the following steps:

1. Creating the database table in the target database
2. Setting up the Documaker Server INI configuration for the connection to that database with the table
3. Setting the Documaker DFD (Data Format Definition) to contain all the field values you want mapped from GVM variables to the database table.

Example Database Table Definition

Each row of the table must include a minimum set of columns required by both Documaker Server and Documaker Connector to manage the list of output documents and the document output directory. The table schema must also include additional columns of customer-specific metadata used to archive the documents, none of which is shown below.

This table shows the minimum required columns of the table with their default column names. You can customize these column names and add your own columns in the Documaker Source configuration file. The following list does not include customer-specific metadata to be archived. In an actual implementation, you would expand this with additional columns to define your metadata. Keep in mind this list of columns *must* match the schema of your database table. For more information, see *Using DAL to Output to a Database Table* on page 93.

Column	Type	Description
JOBID	VARCHAR(50)	The Documaker globally-unique job identifier which identifies a grouping of one or more Documaker transactions for a single import run. Imports (XML or V2) can contain one or more transactions. This column's value is also used by default for the root directory folder of the output print stream.
TRANID	VARCHAR(50)	The Documaker transaction identifier for a transaction with one or more Documaker batches (recipient batches). This column's value is also in some implementations to map to the document title or name and for searching should identify the document type and purpose.
BATCHID	VARCHAR (50)	The Documaker batch identifier for a document, usually the same name as the recipient batch plus a counter. For example, BATCH6x2 where the counter is incremented as the specified maximum number of files per folder is reached. This columns value is also used by default to segment the transaction folder into sub-folders for each group of output files.
DOCID	VARCHAR (50)	The globally-unique document identifier.

Column	Type	Description
NAME	VARCHAR (30)	The name of the document.
TYPE	VARCHAR (30)	The type of document.
TITLE	VARCHAR (255)	The title for the document.
AUTHOR	VARCHAR (50)	The author or owner of the document.
SECGROUP	VARCHAR (30)	The security group assigned to the document.
PFILE	VARCHAR (255)	A file URL or path to the document.
STATUSCD	INTEGER	This column contains the status of a document. The following values are supported: 0 – Not yet processed by Documaker Connector (new) 1 – Imported into content management (success) 2 – Import failed (failure) 3 – In process by Documaker Connector (in progress) The default is zero (0)
STARTTIME	TIMESTAMP	A time stamp that indicates at which time the document import process started. This column is updated by Documaker Connector.
ENDTIME	TIMESTAMP	A time stamp that indicates at which time the document import process ended. This column is updated by Documaker Connector.
RESULTDESC	VARCHAR (2000)	A description of the outcome of the import process; updated by Documaker Connector at the time of import. This column contains *Success* if the document import process is successful. Otherwise, it contains a description of the error.
RETENTION	TIMESTAMP	A time stamp that indicates when the document expires and can be removed from the table. This value is updated by Documaker Connector upon a successful import, based on the value of the source.documaker.retention.time configuration property which indicates the number of days after import when the document expires.

Table 4: Minimum DAL Output Database Table

Documaker INI Setup for the Example Database Connection

Documaker Server is configured to access a database and a particular table in the Documaker configuration INI file (FSISYS.INI or FSIUSER.INI). Documaker uses the Open DataBase Connectivity (ODBC) interface API to the database.

Here is an example:

```
; Database connection info
< DBHandler:ODBC >
   Class          = ODBC
   Server         = OracleXE10g
;  SubClass       = ORA
   CreateTable    = No
   CreateIndex    = No
;  Debug          = Yes
   UserID         = ~ENCRYPTED 1-S6rx_NR_wt2hsjXScy0
   PassWd         = ~ENCRYPTED 1-S6rx_NR_wt2hsjXScy0
;  Database Table reference, this case a table named AOR
< DBTable:AOR >
   DBHandler      = ODBC
```

Note The lines that begin with a semicolon (;) are comments and are not processed.

In these example settings, Documaker Server connects via ODBC to a database server called *OracleXE10g* with an encrypted user ID and password. In this database, Documaker Server is configured to use a table called AOR (Archive Of Record).

Note You can find more information on ODBC database access configuration in the Documaker Administration Guide.

Configuring Oracle WebCenter Content

If you are using Oracle WebCenter Content as your content management system, this section provides information on what you need to set up in WebCenter Content to index data coming from your document source, such as Documaker. Although this topic discusses WebCenter Content configuration, it is not a substitute for the WebCenter Content documentation or familiarity with the WebCenter Content configuration requirements, procedures and user interfaces.

Note The information in this topic explains the minimum required WebCenter Content setup through one set of interfaces and provides some example information so a knowledgeable WebCenter Content administrator has a few reference points to understand the requirements in this document.

Oracle WebCenter Content is listed in the product suite under Fusion Applications, Oracle WebCenter Content, Document Management. On the Downloads site it is under OTN, Middleware, WebCenter Suite, WebCenter Content.

A minimal WebCenter Content system for use with Documaker Connector starts with installation of the Oracle Content Server. Documaker Connector was developed using Oracle Content Server 10gR3, but has been tested with later versions, including 11g.

The metadata and documents you create and capture in Documaker and pass through Documaker Connector must have homes in your WebCenter Content system. That is, you must set up your WebCenter Content system with the information fields needed to hold your incoming metadata. You must also make sure there is plenty of space to store the documents you are sending in via Documaker Connector.

WebCenter Content allows metadata fields to be required or optional. For WebCenter Content fields you map to incoming data, Documaker Connector checks the fields provided in the database records to make sure the required fields are present. If a required field definition is missing from the table, Documaker Connector logs this information and stops.

By default, the names of the columns in the incoming table from Documaker are used to match the data to WebCenter Content property (field) names. Not all the table columns need be mapped to information fields in WebCenter Content. Many of the table columns are used internally by Documaker and the Documaker Source modules to manage the table itself, remove old records which have been processed, and to keep track of incoming documents before they are processed into WebCenter Content.

Within WebCenter Content, use the Configuration Manager to set up the WebCenter Content information fields for the incoming table columns you want to capture. In the example below, information fields have been established for these example custom fields coming in from Documaker:

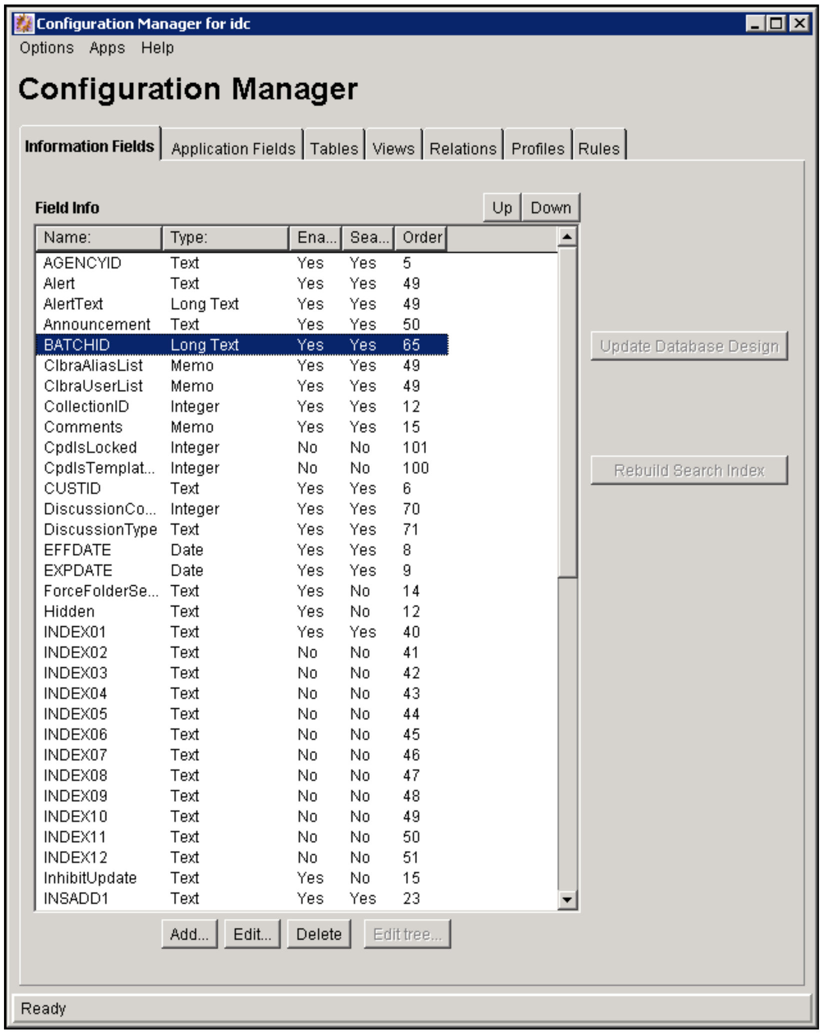
AGENCYID	BATCHID	CUSTID
EFFDATE	EXPDATE	INDEX01, ... INDEX12
INSADD1	INSADD2	INSCITY
INSDOB	INSFNAME	INSLNAME
INSPHONE	INSSTATE	INSZIP

JOBID	KEY1	KEY2
KEYID	POLNUM	RUNDATE
TRANCODE	TRANID	

Of the columns shown in Table 4, *Minimum DAL Output Database Table*, on page 42, these columns are mapped by their default names to the indicated WebCenter Content metadata fields:

- JOBID
- TRANID
- BATCHID

BATCHID, for example, is highlighted in the following Configuration Manager window. The remainder of the custom fields must be handled by adding more columns to the minimum set of columns shown in *Minimum DAL Output Database Table* on page 42. You can find an example of the DDL that does this in *Sample Applications and Files* on page 78. In this example, all the fields shown previously are mapped by default using the same name for both the database table column and the WebCenter Content information field:



Name:	Type:	Ena...	Sea...	Order
AGENCYID	Text	Yes	Yes	5
Alert	Text	Yes	Yes	49
AlertText	Long Text	Yes	Yes	49
Announcement	Text	Yes	Yes	50
BATCHID	Long Text	Yes	Yes	65
ClbraAliasList	Memo	Yes	Yes	49
ClbraUserList	Memo	Yes	Yes	49
CollectionID	Integer	Yes	Yes	12
Comments	Memo	Yes	Yes	15
CpdIsLocked	Integer	No	No	101
CpdIsTemplat...	Integer	No	No	100
CUSTID	Text	Yes	Yes	6
DiscussionCo...	Integer	Yes	Yes	70
DiscussionType	Text	Yes	Yes	71
EFFDATE	Date	Yes	Yes	8
EXPDATE	Date	Yes	Yes	9
ForceFolderSe...	Text	Yes	No	14
Hidden	Text	Yes	No	12
INDEX01	Text	Yes	Yes	40
INDEX02	Text	No	No	41
INDEX03	Text	No	No	42
INDEX04	Text	No	No	43
INDEX05	Text	No	No	44
INDEX06	Text	No	No	45
INDEX07	Text	No	No	46
INDEX08	Text	No	No	47
INDEX09	Text	No	No	48
INDEX10	Text	No	No	49
INDEX11	Text	No	No	50
INDEX12	Text	No	No	51
InhibitUpdate	Text	Yes	No	15
INSADD1	Text	Yes	Yes	23

CONFIGURING DOCUMAKER CONNECTOR

Once you have configured Documaker to generate the data you want and set up the destination content management system to accept and store that data, you can configure Documaker Connector to pass that data from the source to the destination. Documaker Connector needs to be given your choices of source (such as Documaker) and destination systems (such as Oracle WebCenter Content) as well as some operational parameters that are independent of the source and destination systems.

Your source system is the source of your data and documents. This will normally be Documaker, as described in this document. The destination is the system where your documents will be archived. For each system there must be a connector component which is designed to connect to and exchange data with that source or destination and you have to tell the connector application which components to use.

You do this by providing the names of the proper components. Some source and destination components are provided with Documaker Connector, including the Oracle Documaker Source, the Oracle WebCenter Content Destination, and some testing and example components.

USING THE PROPERTIES FILE

Documaker Connector and the standard Oracle interface components use a single common properties file for all configuration settings, described below. Custom source or destination components may be configured differently, but the standard components described in this document all use this file for their settings. If you are developing a custom component, you should also use this means of configuration.

You can enter the name of this properties file on the command line via the -config parameter. The default file name is *connector.properties,* which must be located in the execution directory if you omit the -config parameter.

This file is referenced in the dm_connector_svc.files by uncommenting it and commenting out the rest. The targeted file for use is based on the source and destination combination.

Files	Description
dmkr-ucm-conn.properties	Use dmkr-ucm-conn.properties if the source is dmkr and destination is ucm
mock-mock-conn.properties	Use mock-mock-conn.properties if the source is mock and destination
mock-file-conn.properties	Use mock-file-conn.properties if the source is mock and destination
batch-ucm-conn.properties	Use batch-ucm-conn.properties if the source is batch and destination is ucm

The configurations are contained in the properties file support:

- General Connector configuration
- Source component configuration
- Destination component configuration

The first topic, General Connector Configuration, provides parameters for the connector application which are independent of the source and destination, including identifying which source and destination components to use. The other topics are properties entirely dependent on the particular components chosen in the first topic. This document describes these components:

Sources	• Documaker, using the AOR table • Mock, which generates fake documents and metadata • BatchLoader, which works with the WebCenter Content Batch Loader utility files
Destinations	• Oracle WebCenter Content/Content Server • Oracle WebCenter Content Imaging (formerly Imaging and Process Management (IPM)) • Mock, which accepts documents and metadata and discards them • FTP, which sends documents to a receiving FTP site • File, writes documents into folders in a local file system • AssureSign, submits documents to the AssureSign e-signature service

Note in the following tables, that some of the property names are quite long. To save space in the tables, some groupings of properties that start with common prefixes are listed without the prefix. The common prefix is given above the group of properties in a separate line in the table labeled *Property Name Prefix*. These properties are written starting with a period (.), which is also shown as part of the prefix as a reminder that the property name includes the prefix shown above it. A single period is always used between the prefix and the rest of each property name when it is used in the file.

Note There are *mock* source and destination components that provide *fake* documents and data or let you send them to *nowhere* as a destination. You can use these together to test the installation of Documaker Connector or individually to test a specific source or destination. For example, you can set up the mock destination to test the Documaker source configuration without really archiving the test documents.

Standard Source Configuration Properties

The following list of properties is available to any source implementation. Any additional properties needed by the implementation should be documented in the implementation's guide.

Name	Description	Default
source.name	The fully qualified name of the source implementation	-
source.administration.name	The fully qualified name of the SourceAdministration implementation	-
source.count	The number of instances of the source implementation to create	1
source.max.records	The maximum number of documents to return when the getMetaData method is called	1
source.administration.cleanupwait	The number of seconds between source system cleanup calls.	10
source.import.delete.imported.files	Delete the imported files from the file system.	True
source.import.delete.imported.files.count	The number of files to be deleted during each cleanUp call.	50
source.persistence.path	The directory path to contain any result data that cannot be updated in the source system.	-

Destination Configuration Properties

The following list of properties is available to any destination implementation. any additional properties needed by the implementation should be documented in the implementation's guide.

Name	Description	Default
destination.name	The fully qualified name of the destination implementation	-
destination.administration.name	The fully qualified name of the DestinationAdministration implementation	-

destination.active.wait	The number of seconds to wait for the destination system to return as active	10

Handling Passwords in the Property File

For all properties in Documaker Connector, you can encrypt the property value by writing the value with a prefix of ~ENCRYPT. This is particularly useful for passwords, so the property file is not left with an unprotected password in it. Documaker Connector does this automatically if the property...

- File is not write-protected by the connector application
- Value is written with a prefix of *~ENCRYPT*

If Documaker Connector sees *~ENCRYPT* when it reads the property file, it encrypts the property value and then writes that encrypted value back to the file with a prefix of *~ENCRYPTED*.

For example, if the property file contains this line:

```
source.documaker.password=~ENCRYPT oracle
```

When you run Documaker Connector, it changes the line to read:

```
source.documaker.password=~ENCRYPTED 1-S6rx_NR_wt2hsjXScy0
```

Note that there is no way to make Documaker Connector to decrypt an encrypted property value string.

Note This applies to the Documaker Connector engine and *all* the sources and destinations you set up to have their configurations stored in the main Documaker Connector properties file. It is possible that custom code could handle password protection in another way.

General Connector Configuration Properties

These properties apply to the Oracle Documaker Connector application, regardless of what document source and destination is configured. These items include the choices of source and destination components.

Most of the core application properties configure the common logging service. These properties let you specify the amount information that is logged (this affects performance), where the log files are written, how the log files are named, and how the log is split into multiple files over time.

When you run Documaker Connector in Server mode from the command line, it can be controlled by another copy of itself running in *Commander* mode. The Commander mode copy sends control messages across the network to the server mode copy. To prevent unauthorized or accidental control messages from being processed, a password may be required. These properties control the port on which the server Connector listens for commands and the password used to validate incoming commands on the port.

The *channel.count* controls the number of parallel channels Documaker Connector creates and manages. Using multiple channels allows processing to scale within a single Documaker Connector application. Each channel uses a copy of the source and destination which are all created and initialized separately. Consequently, each channel creates separate connections to the source and destination systems.

If run in server mode, each Documaker Connector channel loops continuously, calling its source until no more documents are available to be processed. It then sleeps for *engine.datarequest.wait* seconds and tries again to import documents.

Complete Property Name	Description	Default
General Connector Configuration Properties		
log.level	Sets the logging level to either FATAL, ERROR, WARN, INFO, or DEBUG.	WARN
log.destination	Specifies where the log is written, either CONSOLE or FILE.	CONSOLE
log.path	Specifies the base log file name.	connector.log
log.maxsize	Specifies the maximum log file size in kilobytes.	100 (Kb)
log.history	Specifies the number of rolled-over history log generations to retain.	One (1)
command.port	Specifies the command channel port number.	23232
command.password	Specifies the password which must be provided by a client with each command request. See *Handling Passwords in the Property File* on page 50 for more information.	none
channel.count	Specifies the number of document import channels. Each channel has a source and destination component created for it.	One (1)
engine.datarequest.wait	Specifies the number of seconds to wait after an empty data request is received.	Five (5)
source.name	Specifies the Java class name for the source component.	none
destination.name	Specifies the Java class name for the destination component.	none
[id.]phase.name	Specifies the Java class name for a phase listener you want to include in the processing cycle.	none

Table 5: General Connector Configuration Properties

Phase Listeners

Documaker Connector's imports process consists of several phases. You can include additional functionality at each phase using a *phase listener*. A phase listener is an add-in to Documaker Connector processing that supplements the basic transfer and import operation handled by the source and destination.

When you configure a phase listener, it automatically runs at all the appropriate points in the process. You can configure any number of phase listeners, as long as what each of them does is compatible with the others.

To configure multiple phase listeners, you must assign an ID to each phase listener except the first. The phase listener with no ID is considered the default, although you assign an ID to all of them if you prefer. Assigning IDs gives you a way to refer to each phase listener as you specify the properties to configure them.

Note You can include any alphanumeric character in an ID.

For example, you might configure two phase listeners that work independently. Imagine that both of them have an output.directory property where they write results. You can configure these phase listeners to use different directories, as shown here:

```
alpha.phase.name = alpha’s classname
bravo.phase.name = bravo’s classname
alpha.output.directory = \alphaDir\
bravo.output.directory = \bravoDir\
```

Periodic Process Properties

While the Documaker Connector application runs, there may be work that can be done on a time available or periodically scheduled basis, rather than as part of transfer and import processing. You can configure these periodic processes independently of both the source and destination, but configuring these processes is typically handled when you configure either the source or the destination and the applicable settings are packaged with them. They are configured with the *periodic.process* settings.

Each periodic process can be replicated into multiple copies. Together, these copies are called a *periodic process collection*. If you have multiple, different processes to run, you create multiple collections.

To configure multiple periodic processes, you must assign an ID to each collection except the first. A collection with no ID is considered the default, although you can assign an ID to all of them if you prefer. Assigning IDs gives you a way to refer to each periodic process collection as you specify the properties to configure them.

An example of a periodic process is the one which deletes the source documents after they are imported using the Documaker source. You may want to leave the documents in the source location for a period of time so the import process can be verified as successful. After this time period and long after the import processing was completed, you can have a periodic process delete the document files. This process is included in the Documaker source jar file and can be configured as shown here to run one copy (instance.count=1), forever (repetition.count=0), or every hour (repetition.wait=3600000):

```
   periodic.process.collection.instance.class = 
oracle.documaker.ecmconnector.documakersource.DocumakerSourceProcess
   periodic.process.collection.instance.count = 1
   periodic.process.repetition.count = 0
   periodic.process.repetition.wait = 3600000
```

Property	Description	Default
Periodic Process Properties Property Name Prefix: periodic.process.		
.collection.instance.count	The number of instances of the periodic process class to be created for the collection.	One (1)
.collection.instance.class	The name of the Java class to be run by the process collection.	none
.repetition.count	The number of times the periodic process should execute. Set to zero (0) for infinite repetitions.	One (1)
.repetition.wait	The wait time in milliseconds between iterations of the periodic process.	5000 (five seconds)

Table 6: Periodic Process Properties

General Source Configuration Properties

Some properties are expected to be common to multiple sources or destination implementations. It is, however, up to any particular source or destination as to whether the property is applicable and how it is specifically implemented. Those properties are defined and described in this topic, so components that choose to implement them can share a common definition. The description of each source or destination which implements these properties should include these properties if they implement them.

Complete Property Name	Description	Default
Source Component General Configuration Properties Property Name Prefix: source.		
.import.delete.imported.files	It is up to each source component to decide if this property is to be used. True indicates Documaker Connector should delete the imported files after they have been successfully imported. This occurs on a time-available basis in a background thread since the batches can be large. How many the thread does at a time is determined by the import.delete.imported.files.count setting. If you set this to False, you must manually delete these files.	True
.import.delete.imported.files.count	It is up to each source component to decide if this property is to be used. Maximum number of files to delete in a single cycle.	50
.persistence.path	The directory which contains any result data that could not be updated to the source system. If the source component cannot relay all the results to the source system (such as the database) this directory is used to save the data if Documaker Connector is shut down.	none

Table 7: Source Component General Configuration Properties

Documaker Source Configuration

You can configure Documaker Server, as discussed *on page 37*, to write records to an Archive of Record (AOR) database table and write documents such as PDF files into a disk directory. This source component works with that configuration and performs these tasks:

- Reads records from the database table
- Passes the metadata and document data to the Documaker Connector destination you specified
- Posts status updates into the database table to note the progress of the archive process and then later to mark it as complete

Note This component is not used with Documaker Document Factory and the Archiver, which supply their own document source.

The Documaker source is configured to make a Java DataBase Connectivity (JDBC) connection to an SQL database table. The Documaker DAL scripts provided in the banner processing configuration for Documaker write a record into this table for each document. The Documaker source queries the table, as described in *Configuring Documaker* on page 37. Each record's information is then passed through Documaker Connector to the configured destination and the destination is told to import the document. The status result is returned to the Documaker source, so it can update the database table.

Since you can configure the Connector application to create multiple channels between the source and destination, the Documaker source can create a pool of database connections from which each channel can draw a connection as needed.

A connection is used to query the document records and then later to set the status results after the import is complete. For optimal performance, the number of channels and database connections should be the same. Most of the Documaker source configuration properties set the characteristics and behavior of this database connection pool.

Configure the use of the Documaker source with:

```
source.name=oracle.documaker.ecmconnector.documakersource.DocumakerS
ource
periodic.process=oracle.documaker.ecmconnector.documakersource.Docum
akerSourceProcess
```

Property Name	Description	Default
Database Connection Properties Property Name Prefix: source.documaker.		
.driver.name	The JDBC driver name to use.	
.url	The JDBC database URL.	
.username	The JDBC account name used for authentication to the database.	
.password	The JDBC password used for authentication to the database. See *Handling Passwords in the Property File* on page 50 for more information.	

Property Name	Description	Default
.table	Sets the JDBC database table name.	
.timeout.seconds	The JDBC connection timeout interval, in seconds.	0
.connection.property.*name*	The JDBC custom connection property's *name*. Used for connections to databases which require connection properties beyond those listed above. Each value is passed to JDBC paired with each *name* listed.	none
Database Connection Performance Tuning Properties		
.pool.size	The number of database connections to create in a pool.	25
.max.active.connections	The maximum active connections in the connection pool.	50
.min.idle.connections	The minimum idle connections in the connection pool.	15
.max.idle.connections	The maximum number of database connections left idle in the pool. Additional idle connections are closed.	25
.max.wait.seconds	The maximum number of seconds to wait for a connection from the connection pool if one is not immediately available.	30
.max.open.prepared.statements	The maximum number of opened prepared SQL statements to keep for the connection pool.	15
.min.evictable.idle.time.millis	The number of milliseconds a connection must be idle before it is cleaned up.	60000 (10 minutes)
.time.between.eviction.runs.millis	The number of milliseconds between idle connection cleanup.	300000 (5 minutes)
.tests.per.eviction.run	The maximum number of connections tested for idle each iteration.	25
.test.on.borrow	Indicates whether to test connection objects when obtaining a connection.	True
.test.while.idle	Indicates whether to test connection objects while idle.	True
.validation.query	The query string used to test if a connection is alive. This query is made if the source.documaker.test.on.borrow property is True. A default string is created if this property is left empty. The default refers to the source.documaker.table attribute shown in the Database Connection Properties table. If, however, a query string is provided, it is used without modification by the Documaker Source.	*select count(1) from tableName*
.max.records	The maximum number of records that should be retrieved from the Documaker AOR table as a group to process.	One (1)
.check.length	If True, the source validates the length of the data retrieved for each column before passing that data to the destination. The length of the data retrieved is checked against the length of the destination metadata for that column and if the length of the data exceeds the space available, a LengthCheckException is thrown.	False
.retention.time	The number of days an imported record will stay in the AOR table before it is removed.	none
.excluded.columns	A comma-separated list of columns in the Documaker AOR table that should be hidden from the destination and not be available as document metadata.	none
.contents.columnname	The name of the column in the AOR table that contains the document contents, if the *..include.file.contents* property is True.	none

Property Name	Description	Default
.row.identifier	The name of the column in the AOR table that contains the row identifier, if the *..include.file.contents* property is True.	none

Table 8: Documaker Source Configuration Properties

The keys to this configuration are the JDBC settings that link Documaker Connector to the database table written by Documaker. The driver name is the class name of a class in the program's CLASSPATH. The Oracle driver, for example, is distributed in the ojdbc14.jar file and is chosen by configuring:

```
source.documaker.driver.name=oracle.jdbc.driver.OracleDriver
```

The driver is directed to the database host with a URL:

```
source.documaker.url=jdbc:oracle:thin:@localhost:1521:xe
```

The URL always starts with *jdbc:* indicating the protocol used. This is followed by a DBMS-indicating preface, such as

- oracle:
- mysql:
- microsoft:sqlserver:
- db2:
- jtds:sqlserver:

The remainder of the URL format is controlled by the specific DBMS driver. For Oracle, the remainder is represented by this format:

```
driver-type:[username/password]@database_specifier
```

Parameter	Description
driver-type	Choose *thin*, *oci*, or *kprb*. This affects the format for the database_specifier property.

username/password	(Optional) Documaker Connector expects this information as separate configuration items:
	`source.documaker.username=username`
	source.documaker.password=password
	See *Handling Passwords in the Property File* on page 50 for more information.

database_specifier	For thin, you can use a string like this one: `@//host_name:port_number/service_name` This connects to an Oracle XE database running on the same computer as Documaker Connector using port 1521. You can find more information on the Oracle JDBC product on the Oracle Technology Network. You can find information for the URL format for your database in your DBMS documentation or on the Internet.

Configure the database table name as shown here:

```
source.documaker.table=tablename
```

The tablename, as well as column names specified elsewhere, can be case-sensitive or case-blind depending on the database settings. One list of column names is provided in the *excluded.columns* property. These columns are hidden from the destination and cannot be used as document metadata. Generally, these columns are used to manage the table content by Documaker or the Documaker Source:

```
source.documaker.excluded.columns=column, column, ...
```

The Documaker Source opens a pool of multiple connections to the database. This pool is shared across all of the instances of the Documaker Source you set up in the Connector parameter:

```
channel.count=number_of_instances
```

The size and behavior of this pool are controlled by several properties. The pool can grow to a maximum number of connections given by

```
source.documaker.max.active.connections=max_number_of_
    connections
```

but the pool starts with the number given in this property:

```
source.documaker.pool.size=initial_number_of_connections
```

If the pool is at its maximum size, a request for a connection will have to wait. The longest it will wait before failing and causing an error is set here:

```
source.documaker.max.wait.seconds=wait_seconds
```

Once the pool is created and connections begin to be used, additional connections are opened and added to the idle pool if the number of idle connections drops below the configured minimum, but only up to the maximum count given above as source.documaker.max.active.connections. Specify the minimum number of idle connections here:

```
source.documaker.min.idle.connections=desired_idle_connections
```

As connections become idle, they remain in the pool until the number of idle connections rises above the configured maximum:

```
source.documaker.max.idle.connections=max_idle_connections
```

The pool is managed by a periodic process which checks these numbers. This process is called *eviction* and runs at a specified interval. At each run, Documaker Connector checks the idle connections to see if they have been idle long enough to be eligible for eviction from the pool — closed. That minimum time is also configurable.

```
source.documaker.time.between.eviction.runs.millis=eviction_
    interval_milliseconds
source.documaker.min.evictable.idle.time.millis=evict_min_idle_
    milliseconds
```

While connections are sitting idle in the pool, they can become *stale* if there is a network problem or the DBMS chooses to close the connection due to a time-out setting, for example. The Documaker Connector Source can maintain these connections and avoid timeout intervals by periodically running test transactions during eviction processing. Because this can be time-consuming, especially if the idle pool is large, you can limit it to a few connections per interval, rather than doing all of them every time:

```
source.documaker.test.while.idle={true | false}
source.documaker.tests.per.eviction.run=number_of_connections_
   to_test
```

Similarly, before a connection is pulled from the pool to use, it can be tested if this property is set:

```
source.documaker.test.on.borrow={true | false}
```

The transaction that is run to check the connection is usually named *SELECT 1 FROM tablename*. You can select a different transaction by setting this property:

```
source.documaker.validation.query=SQL_query
```

Oracle WebCenter Content, Core Capabilities Destination Configuration

This destination archives documents into Oracle Universal Content Management/ Content Server. It supports pushing metadata and documents into a flat WebCenter Content archive via streaming the document content as well as having WebCenter Content pull the document in from a common disk location. WebCenter Content folders are not supported. Direct connection to Content Server is supported via the Remote IDC (RIDC) protocol.

Configure the use of the Oracle WebCenter Content destination with:

```
destination.name=oracle.documaker.ecmconnector.ucmdestination.
UCMDestination
```

Property Name	Description	Default
UCM Destination Properties Property Name Prefix: destination.ucm.		
.username	The WebCenter Content/Content Server user name.	
.password	The WebCenter Content/Content Server password. See *Handling Passwords in the Property File* on page 50 for more information.	
.connectionstring_#	The possible connection strings to try in order. The appended number indicates desirability, with zero (0) being the most desirable.	
.column.map.*COLUMNNAME*	Maps the source column *COLUMNNAME* to a WebCenter Content field name (property value).	
.importmethod	Specifies whether to import the source document by file (1) or by stream (0).	one (1)

Table 9: WebCenter Content, Core Capabilities Destination Configuration Properties

WebCenter Content Data Mapping

Use of the destination.ucm.column.map.COLUMNNAME is straightforward. The placeholder COLUMNNAME is replaced by an AOR table column-name and the value assigned is the name of the Content Server metadata field. Both names are unique in their namespaces. Here are some examples:

```
destination.ucm.column.map.NAME=dDocName
destination.ucm.column.map.AUTHOR=dDocAuthor
destination.ucm.column.map.TITLE=dDocTitle
destination.ucm.column.map.TYPE=dDocType
```

A single COLUMNNAME can be mapped to multiple WebCenter Content fields by listing it on multiple lines. It is an error to attempt to map multiple incoming COLUMNNAME fields to the same WebCenter Content metadata field name.

Oracle WebCenter Content, Imaging Destination Configuration

Formerly known as the Oracle Imaging and Process Management (IPM) product, WebCenter Content, Imaging adds another set of capabilities to WebCenter Content, Core Capabilities (formerly UCM). As such, it has its own set of interfaces which you must use to import documents into the imaging system, rather than directly into the underlying core capabilities system.

Note This destination was developed before the product naming change, so it is referred to as the *IPM destination*.

Configure the IPM destination with:

```
destination.name =
oracle.documaker.ecmconnector.ipmdestination.IPMDestination
```

This table shows the configuration properties:

Property Name	Description	Default
IPM Destination Properties Property Name Prefix: destination.ipm.		
.connection.string	The string used to reach the proper WebCenter Content, Imaging system.	none
.user.name	The WebCenter Content, Imaging user name to use to connect.	none
.password	The WebCenter Content, Imaging password for the connection.	none
.application.id	The WebCenter Content, Imaging application ID.	none
.application.name	The WebCenter Content, Imaging application name.	none

Table 10: WebCenter Content, Imaging Destination Configuration Properties

Other Sources and Destinations

Sources

Use these properties to configure Documaker Connector with other sources.

BatchLoaderSource Properties

The BatchLoaderSource is a sample source component you can use with the WebCenter Content batch loader script files (although the only action supported is import). The BatchLoaderSource reads import records from batch files that it gets from a batch queue file. These records contain the data necessary to import a document into a destination (specifically, the WebCenter Content, but other destinations could be used with the proper batch files).

Configure the use of the BatchLoaderSource with:

```
source.name=oracle.documaker.ecmconnector.batchloadersource.
BatchLoaderSource
periodic.process=oracle.documaker.ecmconnector.batchloadersource.
BatchLoaderProcess
```

Property	Description	Default
source.batchloader.batchfile	The name of a batch file to be read for import records. This is mainly used when you are running the Connector in Singleton mode.	none
source.batchloader.batchqueuefile	The name of the batch queue file that contains the list of batch files. As the files are processed, they are removed from this file. New files can be added at any time. This is mainly used when you are running the Connector is running in Server mode.	none
source.import.delete.imported.files	Used as specified. For more information, see Table 5 on page 52.	False
source.import.delete.imported.files.count	Used as specified. For more information, see Table 5 on page 52.	Zero (0)
batchloader.source.max.records	The maximum number of records to read from the batch files before processing starts.	One (1)
source.batchloader.errordirectory	The directory where error files will be written for each batch file. The error file names will be in one of these formats: `<batch file name>.SOURCEERRORS` for errors in the batch file itself (bad data or action) or `<batch file name>.IMPORTERRORS` for errors from the import attempts.	none
source.persistence.path	See *General Source Configuration Properties* on page 54.	

Table 11: BatchLoaderSource Properties

DocumentDataSource Properties

There are no properties for this source component. This component is provided only for use in custom applications written to use the Connector. It cannot be configured and used as a stand-alone source component.

Mock Source Properties

Use these properties to set up a mock source component. The mock source lets you test your implementation without having actual incoming documents. It generates fake documents and metadata based on its configuration and can also generate errors to test how Documaker Connector and the destination handle errors.

You can use it to configure and test the destination before your source is ready to generate test documents. Developers can also use it to generate test data when developing a destination.

Configure the use of the mock source with:

```
source.name=oracle.documaker.ecmconnector.mocksource.MockSource
```

Property	Description	Default
Mock Source Properties Property Name Prefix: source.mock.		
.empty.lists.allowed	Determines if the empty document lists can be returned from the acquireDocuments method call.	False
.author	The author property for each mock document data.	none
.title	The title property for each mock document data.	none
.type	The document type property for each mock document data.	none
.secgroup	The security group property for each mock document data.	none
.filepath	The file path property for each mock document data.	none
.import.errors	Specifies whether to generate random import errors.	False
.import.error.threshold	The threshold value above which an import error is generated.	75 (out of 100)
.runcount	The number of import cycles to execute before the source closes itself. This is useful for testing in Singleton mode.	Zero (0), which means no limit.

Table 12: Mock Source Properties

DESTINATIONS

Use these properties to configure Documaker Connector with other destinations.

AssureSign Destination Properties

This destination submits/launches documents in the AssureSign system for digital signatures. You must have an AssureSign account.

Note Visit this website for more information:

http://www.assuresign.com/

Configure the use of the AssureSign Destination with:

```
destination.name=oracle.documaker.ecmconnector.assuresigndestination
.AssureSignDestination
```

Property	Description	Default
AssureSign Destination Properties Property Name Prefix: destination.assuresign.		
.username	The AssureSign account name you want to use to launch documents.	none
.contextidentifier	The AssureSign account context identifier.	none
.javax.net.ssl.trustStore	The trust store to use with requests to the AssureSign service.	none
.javax.net.ssl.trustStorePassword	The password for accessing the trust store that contains the AssureSign certificate.	none
Note: The trustStore and trustStorePassword properties override the defaults specified by the JVM running the Connector. You may want to erase these values all together. To do this, set the value for either or both of these properties to “*(erase)*”. You must specify either the destination.assuresign.template.identifier or the destination.assuresign.template.name. The identifier takes precedence		

These values configure the destination component's proxy host values. These may be needed to access the AssureSign service. If not, they should not be configured.

Property	Description	Default
.https.proxyHost	The name of the proxy host necessary to read the AssureSign service.	none
.https.proxyPort	The port number at which to access the proxy host.	none
.https.proxyUser	(Optional) the user name for accessing the proxy host.	none
.https.proxyPassword	(Optional) The password for accessing the proxy host.	none
.template.identifier	The default template identifier you want to use when launching each document.	none
.template.name	(Optional) The default template name used to determine the template ID.	none

You must specify either the destination.assuresign.template.identifier or the destination.assuresign.template.name. The identifier takes precedence. Each template parameter has a name and a value. An identifier is prepended to this prefix and also to the destination.assuresign.template.parameter.value property to link the two. Here is an example:

Property	Description	Default
.template.parameter.name	The prefix defining a template parameter name.	none

.template.parameter.value	The prefix defining a template parameter value. Here is an example:
	`destination.assuresign.template.parameter.name.sig1= Signatory 1 Name`
	`destination.assuresign.template.parameter.value.sig1=John Smith`
		none

.import.method	This property specifies the method for acquiring the document's contents. You can enter *file* or *stream*.	none
.agreement.statement	The default agreement statement text you want added to the standard AssureSign agreement text. Choose from *file* or *stream*.	file
.compliance.statement	The default compliance statement text you want added to the standard AssureSign compliance text.	none
.extended.disclosures	The default extended disclosures text you want added to the standard AssureSign extended disclosures text.	none

You can override many of these values in the document data you provide with each import request.

Property	Description	Default
.document.type	This property specifies the type of document to be imported, such as PDF, DOC, or DOCX.	none
.order.number	(Optional) The order number associated with the launched document.	none
.document.name	The document's name.	
.expiration.date	(Optional) The expiration date for signatories of the launched document. See the AssureSign documentation for more information.	none

The system returns these values in the document's data after an import or launch:

Property	Description	Default
.document.identifier	The AssureSign identifier for this document.	none
.authorization.token	The authorization token associated with a particular document in the AssureSign system.	none

Table 13: AssureSign Destination Properties

Silanis Destination Properties

This destination submits/launches documents in the Silanis system for digital signatures. You must have an Silanis account.

Note In order to process documents enabled for Silanis electronic signing you will need to activate an eSignLive account with Silanis. This release of Documaker entitles you to a free 30 day account with Silanis. You can activate your account by clicking here http://secure.silanis.com/OracleDocumaker.html.

See Silanis Tutorials for more information.

Configure the use of the Silanis Destination with:

```
destination.name=oracle.documaker.ecmconnector.silanisdestination.
silanisDestination
```

.

Property	Description	Value
destination.silanis.username	Username to access the Silanis System	Username
destination.silanis.password	Password to access the Silanis System	Password
destination.silanissign.proxyhost	The proxy server (optional)	Hostname
destination.silanissign.proxyport	The proxy server port (optional)	Port
destination.silanissign.url	The Silanis signing web service URL	URL
destination.silanissign.referencetext	The reference line in the email generated by Silanis.	Reference Text

Note The Silanis system expects PDF documents so be sure the batch is configured with the correct print and MIME type.

File Destination Properties

Use the File destination to write output documents into a file system directory. By default, the system creates subdirectories in the specified base directory based on each batch identifier.

Note The File destination includes functionality previously in the FilePlus destination.

Optionally, you can have the system write separate *side-car* files, which contain some or all of the metadata for each document, to a destination directory you specify. The metadata files are requested by specifying a pattern for their names.

The default output file names and the optional side-car metadata file content can be formatted and controlled using a simple template tag-substitution language. The template tag values are drawn from each document's metadata and the batch identifier (BATCHID).

Note See *Using templates* on page 67 for more information.

Once the system writes a document into the destination directory, its new file name is added to the document's metadata as the value of this name:

```
destination.file.generated.file.name
```

If you specified a side-car metadata file, its name is added using this name:

```
destination.file.generated.side.file.name
```

Using this destination as an intermediate stop can be a convenient aide in constructing a bridge to another destination that is not directly supported. Stand-alone, possibly custom, import programs can pick up the files and process them into an archive system.

Configure the use of the File destination with:

```
destination.name=oracle.documaker.ecmconnector.filedestination.FileD
estination
```

Property	Description	Default
File Destination Properties Property Name Prefix: destination.file.		
.base.directory	The root directory where the output document files will be placed. The system creates this directory if it does not exist.	none
.side.base.directory	The root directory where the metadata files will be placed. The system creates this directory if it does not exist.	Same as base.directory
.name.pattern	The pattern for the destination file name. This pattern can use the document metadata items as well as the batch ID. If this property is not provided, the source document name is used.	source document name
.side.name.pattern	(Optional) The metadata file name pattern. If you do not want the metadata file included in the output, leave this property empty. If this property is provided, you must also provide the.template or.template.path.	none
.subdirectory.pattern	The pattern for the destination file subdirectory to be used or created under the.base.directory location.	${BATCHID}
.side.subdirectory.pattern	The pattern for the metadata file subdirectory to be used or created under the.side.base.directory location.	${BATCHID}
.template.path	The path to a template file for the side metadata file to be written with each output document.	none
.template	Used if the.template.path property is unspecified. This is the metadata file contents template as a single string.	none

Table 14: File Destination Properties

Using templates

The templates referenced in the File Destination Properties use simple tag substitutions. You can use the value of any file metadata property in a template expression by enclosing the property name in braces and preceding it with a dollar sign. This creates a tag such as the one shown here:

```
${name}
```

This tag is then replaced with the value of the specified property for the document.

All incoming property values are converted to strings of characters when referenced in a tag. The substr function provides a way to use only a portion of a value string. The substr function and parameters replace the property name in the tag and are written this way:

```
${substr(name, start[,end])}
```

Parameter	Description
name	The name of the parameter to substring.
start	The zero-based position of the first character to use.
end	The optional one-based position of the last character to use. The default is the end of the value.

For example, if an incoming property name is *USER_ID* and the value is *user:John Doe*, then the following tag produces the value *John Doe*:

```
${substr(USER_ID,5)}
```

Likewise, this tag becomes *John* when processed:

```
${substr(USER_ID,5,9)}
```

Date (time) values retain their type as a date when carried in parameters. When referenced as a template tag, however, the date is converted to a string. The default format of this conversion is influenced by the locale and date settings on the system running Documaker Connector.

Use this syntax to control the date and time format:

```
${name?string.dateformat[_timeformat]}
```

Parameter	Description
name	The name of the date parameter to format
xxxformat	Specify one of these options: short, medium, long, full, short_long, or long_short. short : 4/27/12 12:34 PM medium: Apr 27, 2012 12:34:56 PM long: April 27, 2012 12:34:56 PM EDT full: Friday, April 27, 2012 12:34:56 PM EDT short_long: 4/27/12 12:34:56 PM EDT long_short: April 27, 2012 12:34 PM

How everything looks exactly is affected by your local system settings. Instead of using the default formats, you can specify the exact format using the Java date format syntax for *pattern*:

```
${name?string(pattern)}
```

Here are some examples:

```
${myDate?string("yyyy-MM-dd HH:mm:ss zzzz")}
    2012-04-27 12:34:56 Eastern Daylight Time
${myDate?string("EEE, MMM d, ''yy")}
    Fri Apr 27, '12
${myDate?string("EEEE, MMMM dd, yyyy, hh:mm a '('zzz')'")}
    Friday, April 27, 2012, 12:34 PM (EDT)
```

Documaker Connector uses an open source library called FreeMarker to provide much of the template functionality. For more information, see the FreeMarker documentation located at this website:

http://freemarker.sourceforge.net/docs/ref_builtins_date.html

Keep in mind you can only use the capabilities of the library that are supported by our implementation within Documaker Connector. For example, parameter values are all strings or dates. All other types are converted to strings.

FTP Destination Properties

You can use this destination component in a limited production environment or for debugging purposes. The FTP Destination configures an FTP site as the document destination. Documents are copied by batch to the receiving site as an archive.

Configure the use of the FTP Destination with:

```
destination.name=oracle.documaker.ecmconnector.ftpdestination.
FTPDestination
```

Property	Description	Default
FTP Destination Properties.		
.server	The FTP server name.	none
.username	The user name needed to log onto the FTP server.	none
.password	The password needed to log into the FTP server. You can encrypt the password using the ~ENCRYPT function.	none
.base.directory	The root directory on the FTP server where files should be copied.	none
server.protocol.type	Identifies the communication protocol, either FTP or Secure FTP (SFTP). Default value is "FTP".	none

Table 15: FTP Destination Properties

Mock Destination Properties

Use these properties to set up a mock destination. A mock destination simply accepts the incoming documents and discards them. Use this destination to test your source configuration with test documents before your destination is ready to receive documents or to test without cluttering your destination system with test documents.

Developers building a custom source can use this destination to discard test data. You can also set this destination to generate random errors to test up-stream error processing while developing a custom source.

Configure the use of the Mock Destination with:

```
destination.name=oracle.documaker.ecmconnector.mockdestination.
MockDestination
```

Property	Description	Default
Mock Destination Properties Property Name Prefix: destination.mock.		
.import.errors	Specifies whether to generate random import errors.	False
.import.error.threshold	The threshold value above which an import error should be generated.	75 (out of 100)

Table 16: Mock Destination Properties

Chapter 4

Running Documaker Connector

This chapter describes how to run Documaker Connector. It covers these topics:

OVERVIEW

You can run Documaker Connector in these different modes:

- (*Singleton* mode) As a batch-import utility application which processes a set of incoming documents and then terminates when there is no more data to process.
- (*Server* mode) As a continuous batch-import daemon or service application which periodically polls the source for new incoming data, processes any available data and then sleeps until the next polling interval. This can be running as a Windows Service or as a UNIX daemon or as some other faceless background task.
- (*Commander* mode) As a controller application sending commands to another copy of the program running in the server mode.

These modes are described in the following topics. Command line execution is generally done with a script or batch file to lessen the complexity of the Java command line. The basic format is:

```
java [-cp classpath] [systemparams] mainclass [parameters]
```

Item	Description

classpath	*classpath* is installation dependent. Generally, it includes three sets of Java JAR files in a delimited list (using a semicolon on Windows and a colon elsewhere as the delimiter):
	Here are the general Java run-time support classes:
	`log4j-1.2.15.jar`
	`commons-beanutils.jar`
	`commons-dbcp-1.2.2.jar`
	`commons-pool-1.3.jar`
	`ojdbc14.jar`
	`DdlUtils-1.0.jar`
	`commons-lang-2.3.jar`
	`commons-collections-3.2.jar`
	`Jakarta-oro-2.0.8.jar`
	Here are the support classes for the core application, including the configurable sources and destinations:
	`Connector.jar`
	`DocumakerSource.jar`
	`UCMDestination.jar`
	Here are the support classes for the configurable sources and destinations:
	`commons-codec.jar`
	`commons-httpclient-3.1.jar`
	`commons-logging-1.1.1.jar`
	`oracle.ucm.ridc-11.1.1.jar`

systemparams	Parameters passed directly to the JVM.
mainclass	*mainclass* controls which version of the program is run. The possible values for *mainclass* correspond to the three modes of execution listed above. It must be one of: • oracle.documaker.ecmconnector.applications.Singleton • oracle.documaker.ecmconnector.applications.Server • oracle.documaker.ecmconnector.applications.Commander
parameters	Application-defined information described below, that depends on the specific mainclass chosen.

Understanding the Modes of Operation

As described above, Documaker Connector has three modes of operation. The mode is selected based on the mainclass specified on the command line:

- oracle.documaker.ecmconnector.applications.*Singleton*

  Batch or *one-shot* mode – In this mode, Documaker Connector creates the configured number of Documaker Source instances, calls each one once to fetch and process a single batch of transactions (if any are available) and then terminates. No socket is opened for commands and there is no subsequent polling of the Documaker Source instances. In the case of the Documaker source, this allows for a single pass through any records in the database table for each configured source instance.

  This is typically used for a non-persistent, static document source, rather than with the Oracle Documaker source. For example, you could use this to read a named input text file. Such a source does not continually receive new documents, so once the configured source of documents (such as the single named input file) is exhausted, there is no need for Documaker Connector to continue to run. This would function similarly to the WebCenter Content BatchLoader application.

- oracle.documaker.ecmconnector.applications.*Server*

  Server or *Normal continuous* mode – Documaker Connector runs until it receives a shutdown command, polling the source instances for documents to process. To receive control commands, Documaker Connector opens a TCP/IP socket and listens for incoming messages. The port number can be controlled by a configuration parameter and a password can be established which must be supplied along with any commands.

- oracle.documaker.ecmconnector.applications.*Commander*

  Command mode – Documaker Connector can also be run solely to send a command to another copy of Documaker Connector running in *Server* mode. You specify the command as a parameter on the command line. When run in this mode, Documaker Connector sends the command to the designated host name and port with the supplied password, if any, and immediately terminates.

Processing Data

For the Singleton and Server modes, you can use the optional *systemparams* argument to point the program to a configuration file. In this case, the *parameters* argument is not used.

You can define the *config* symbol with the full or relative path to the configuration file. The command lines then look like this:

```
java [-cp classpath] [-Dconfig=configpath] mainclass
```

where *mainclass* is the Singleton or Server class reference and *classpath* provides all the necessary support and program classes. If you omit *–Dconfig=configpath*, the program looks for a for a file called *connector.properties* in the execution directory.

The *parameters* argument is not used for these cases. All configuration parameters come from the config properties file.

Controlling a Connector Server Instance

For the Commander mode, there are no *systemparams*. The command line looks like this:

```
java [-cp classpath] oracle.docu…tions.Commander parameters
```

mainclass is replaced by the *oracle.documaker.ecmconnector.applications.Commander* class reference.

Because the Commander mode uses far fewer support classes than actually running a connector workload, a much simpler classpath provides all the necessary support classes. The actual classpath is installation dependent. For the Commander mode, it only needs these JAR files:

- log4j-1.2.15.jar
- connector.jar

The Commander mode opens a connection to another copy of the connector application (the *target* copy) which is already running, presumably in a faceless background mode on the same or another host computer.

It then sends a command to the other program copy. The only available command at this time is the shutdown command, which stops the other program's execution in an orderly way. You specify the location of the other program copy in the command line parameters. Here is a description of the parameters:

Parameter	Description
-host	This is the host IP address or DNS name of the computer running the target copy. The default is localhost.
-port	This is the port on which the target program is configured to listen for commands. The default is 23232.
-pword	This is a password configured on the target required to allow command input. There is no default.
-command	This is the command to send. Only the shutdown command is currently implemented, which is also the default value.

Here is an example of a Commander mode command line on Windows:

```
java -cp log4j-1.2.15.jar;Connector.jar
oracle.documaker.ecmconnector.Commander -host 127.0.0.1 -port 23232 -
pword boogie -command shutdown
```

Since this command line uses defaults, an equivalent command would be to just provide a password:

```
java -cp log4j-1.2.15.jar;Connector.jar
oracle.documaker.ecmconnector.Commander -pword boogie
```

These commands reach out to another copy of the application running in Server mode on the same machine with the default port number and shut it down.

USING A SCRIPT TO RUN DOCUMAKER CONNECTOR

Included in the installations are scripts you can use to start and stop Documaker Connector via a command prompt window. The name of the script varies, depending on the platform.

For this platform	Use
Windows	ecmconnector.bat
UNIX	ecmconnector.sh

These scripts work the same way and have the same parameters. Here is the syntax:

```
ecmconnector (parameters)
```

The parameters include:

Parameter	Description	
-libdir path	(Optional) Sets the Java classpath prefix for where to find all the Documaker Connector JAR files. The default path is ./lib	
-action (type]	Specifies the type of action to perform on this run. You must choose one of these options: start, runonce, or stop.	
	start *(run parameters)*	Runs the ECMConnector in server mode.
	runonce *(run parameters)*	Runs the ECMConnector in singleton mode.
	stop *(terminate parameters)*	Terminates a currently running ECMConnector.

If action type is either *start* or *runonce*, these run parameters are required:

-source *(source)*	Specifies the document source component.
-destination (*destination*)	Specifies the destination component.

The options for source and destination are shown here:

Source	Description	Destination	Description
dmkr	DocumakerSource	ucm	UCMDestination
batch	BatchLoaderSource	file	FileDestination
mock	MockSource	ftp	FTPDestination
		mock	MockDestination

If action type is *stop*, all of these are optional terminate parameters:

Parameter	Description
-pword password	(Optional) Specifies the password needed to access the running instance. The default is no password.
-hostname server_host	(Optional) Specifies the name of the server where the running instance is. The default is localhost.
-port port	(Optional) Specifies the port number at which the running instance accepts commands. The default is 23232.

Here are some examples:

```
ecmconnector.bat -action start -source dmkr -destination ucm
ecmconnector.bat -action start -source mock -destination mock -libdir
ecmconnector.bat -action stop -pword please -libdir ./mylibdir
ecmconnector.bat -action stop -libdir
```

Note `The usage is ecmconnector.bat -action [start/runonce/stop] -source [dmkr/batch/mock] -destination [ucm/file/ftp/mock]`

Appendix A

Sample Applications and Files

This appendix provides examples of the Windows Service Applications and sample setup files:

WINDOWS SERVICE APPLICATION

The dm_connector_svc.exe service application is a launcher/wrapper for Documaker Connector. This application is set up to run as a Windows Service. It can perform the service installation itself or uninstall the service by being run from the command line. If run from the command line, include one of these parameters:

install	Installs the Windows Service and terminates.
uninstall	Removes the Windows Service and terminates.
console	Runs the java application as a console application, for troubleshooting purposes.

Here is an example:

```
dm_connector_svc.exe install
```

If the program is run without parameters, it must run as a Windows Service.

The dm_connector_svc.properties File

The application looks for a dm_connector_svc.properties file and uses the contents to load and run a java application as a service. The service configuration is read from the file:

```
dm_connector_svc.properties
```

The properties in this file and the default values provided in the case of Documaker Connector are shown in this table:

Property Name	**Description**
Oracle Documaker Connector Service Wrapper Properties Property Name Prefix: service.	
.debugging	Set to one (1) to enable debug-level logging to the file dm_connector_svc-service.log.
.jvm.args.length	Count of service.jvm.args.# arguments. The properties starting with the service.jvm.args prefix define the parameters passed to the JVM when it is created. These are not the parameters passed to the main() function in Java (see the service.main prefix items).
.jvm.args.1	First argument to the JVM.
.jvm.args.2	Second argument to the JVM.
.startup.class	Path to the Java class which contains the main() function called to start the Java application.
.path	Directories prepended to the PATH for the service session. The main use of this is to define the JVM used to run the program.
.main.args.length	Count of service.main.args.# arguments. The properties starting with the service.main prefix define the parameters that are passed to the Java main function.
.main.args.1	First argument to the *main* Java class.

Table 17: Connector Service Wrapper Properties

Running Multiple Services

You can set up multiple copies of the executable service program, dm_connector_svc.exe, in the same directory under different names with separate properties file names and they will not interfere with one another. The different copies can run the same or different Java programs, but in this case you would set them to run Documaker Connector. You can use this to set up multiple copies of Documaker Connector running as services for different purposes and with different configurations.

To set up multiple copies, duplicate both the executable service program, dm_connector_svc.exe, and the matching properties file, dm_connector_svc.properties.

Rename the duplicate copies with any name you like, but the two root file names, such as *dm_connector_svc*, must be identical. For example, name the duplicate copies *myservice.exe* and *myservice.properties.* Inside the copied properties file, (*myservice.properties*), change the Windows Service name and description which will be registered when the service is installed:

```
service.name=My Dmkr Connector
service.description=My second copy of the Documaker Connector
```

Also set the name of the Connector properties file that contains your configuration as the second argument to the program:

```
service.jvm.args.2=-Dconfig\=dmkr-ucm-conn.properties
```

The \= (backslash and equal sign) is required to preserve the = (equal sign) when the file is processed.

Example XML Extract File

This example shows you how to use TRN_FIELDS INI control group to map index data in Documaker. The example includes the following XML extract file plus the following setup files:

- *Example Trn_Fields INI Settings* on page 85
- *Example TRNDFDFL.DFD File* on page 86
- *Example RCBDFDFL.DFD File* on page 92
- *Using DAL to Output to a Database Table* on page 93

This example XML file supplies the input variable data to Documaker. This example contains two transactions, each of which generates one or more output documents. Each transaction is an XML document and each starts with the *<?xml...>* header record indicated in bold.

The XML is concatenated into a stream into Documaker. In this example, the *italic* text is mapped data which is used as metadata in the destination system.

```
<?xml version="1.0" encoding="UTF-8"?>
<InterfaceRequest>
   <Header>
      <Key1>DOCCDEMO</Key1>
      <Key2>LIFE</Key2>
      <KeyID>67-875747</KeyID>
      <Run_Date>01-OCT-2008 04:12:58 PM</Run_Date>
      <TRANCODE>NB</TRANCODE>
      <DOCTYPE>LIFE</DOCTYPE>
      <PRODUCT>Foundation Life</PRODUCT>
      <SECGROUP>Archived</SECGROUP>
      <AUTHOR>Steven Doe</AUTHOR>
      <CABINET>CAB1</CABINET>
   </Header>
   <SystemRequest>
      <MessageID>1236474</MessageID>
      <Target>EPOLICY</Target>
      <Target>
         <GO>35235</GO>
         <mode>print</mode>
      </Target>
      <CMD>Print</CMD>
   </SystemRequest>
   <Data>
      <POLICY_NUMBER>67-875747</POLICY_NUMBER>
      <POLICY_ISSUE_DATE>01-OCT-2008 04:12:58 PM</
POLICY_ISSUE_DATE>
      <EFFDATE>01-NOV-2008 12:00:00.00 AM</EFFDATE>
      <EXPDATE>01-NOV-2009 12:00:00.00 AM</EXPDATE>
      <CLASS_OF_RISK>A</CLASS_OF_RISK>
      <STATE_CODE>TX</STATE_CODE>
      <PAYEE>Carl Doe</PAYEE>
      <CUSTID>cjd01</CUSTID>
      <INSURED>
         <PREFIX>Mr.</PREFIX>
         <FNAME>Carl</FNAME>
         <MNAME></MNAME>
         <LNAME>Doe</LNAME>
         <SEX>M</SEX>
         <ADDRESS1>2727 Paces Ferry Road</ADDRESS1>
         <ADDRESS2>Apartment 900</ADDRESS2>
```

```
            <CITY>Atlanta</CITY>
            <STATE>GA</STATE>
            <ZIP>30339</ZIP>
            <BIRTHDATE>15-JUL-1980</BIRTHDATE>
            <INSSSAN>123456789</INSSSAN>
            <DAYPHONE>2148762789778</DAYPHONE>
            <NIGHTPHONE>2148974464</NIGHTPHONE>
            <BIRTHCITY>Anaheim</BIRTHCITY>
            <BIRTHSTATE>CA</BIRTHSTATE>
            <DRIVERSTATE>FL</DRIVERSTATE>
            <DRIVERLICENSE>987987YIU</DRIVERLICENSE>
        </INSURED>
        <AGENT>
            <PREFIX>Mr.</PREFIX>
            <FNAME>John</FNAME>
            <LNAME>Doe</LNAME>
            <ADDRESS1>1100 Abernathy Road</ADDRESS1>
            <CITY>Atlanta</CITY>
            <STATE>GA</STATE>
            <ZIP>30328</ZIP>
            <EMAIL>johndoe@example.com</EMAIL>
            <PHONE>2148582200</PHONE>
            <AgentNo>R98798</AgentNo>
            <CustServPhone>8882637436</CustServPhone>
            <CustServOpenTime>8:00</CustServOpenTime>
            <CustServCloseTime>5:00</CustServCloseTime>
            <CustServTimeZone>eastern</CustServTimeZone>
        </AGENT>
        <POLICY_DATA>
            <PolicyValue>10000000</PolicyValue>
            <PolicyIssueDate>01032005</PolicyIssueDate>
            <PolicyEndDate>01032025</PolicyEndDate>
            <IssueState>GA</IssueState>
            <CostofInsurance>99200</CostofInsurance>
            <CostofInsuranceRate>992</CostofInsuranceRate>
            <CostofInsurance_Option>Level</CostofInsurance_Option>
            <Smoker>N</Smoker>
            <DeathBenefitType>Increasing</DeathBenefitType>
            <AnnualPremium>101900</AnnualPremium>
            <PremiumFrequency>Monthly</PremiumFrequency>
            <PremiumAmount>8492</PremiumAmount>
            <FlatExtra>0</FlatExtra>
            <AdminCharges>2700</AdminCharges>
            <MultipleExtra>0</MultipleExtra>
        </POLICY_DATA>
        <BENEFICIARY>
            <Name>Mary Doe</Name>
            <Relationship>Wife</Relationship>
        </BENEFICIARY>
        <BENEFICIARY>
            <Name>Holly Doe</Name>
            <Relationship>Daughter</Relationship>
        </BENEFICIARY>
    </Data>
</InterfaceRequest>
<?xml version="1.0" encoding="UTF-8"?>
<InterfaceRequest>
    <Header>
        <Key1>DOCCDEMO</Key1>
        <Key2>LIFE</Key2>
        <KeyID>99-456789</KeyID>
        <Run_Date>12-OCT-2008 10:31:12.01 AM</Run_Date>
        <TRANCODE>NB</TRANCODE>
        <DOCTYPE>LIFE</DOCTYPE>
```

```
            <PRODUCT>Foundation Life</PRODUCT>
            <SECGROUP>Archived</SECGROUP>
            <AUTHOR>Carl Doe</AUTHOR>
            <CABINET>CAB1</CABINET>
        </Header>
        <SystemRequest>
            <MessageID>1236474</MessageID>
            <Target>EPOLICY</Target>
            <Target>
                <GO>35235</GO>
                <mode>print</mode>
            </Target>
            <CMD>Print</CMD>
        </SystemRequest>
        <Data>
            <POLICY_NUMBER>99-456789</POLICY_NUMBER>
            <POLICY_ISSUE_DATE>10-OCT-2008 10:31:12.01 AM</
    POLICY_ISSUE_DATE>
            <EFFDATE>01-NOV-2008 12:00:00.01 AM</EFFDATE>
            <EXPDATE>01-NOV-2009 12:00:00.01 AM</EXPDATE>
            <CLASS_OF_RISK>A</CLASS_OF_RISK>
            <STATE_CODE>GA</STATE_CODE>
            <PAYEE>Steven Doe</PAYEE>
            <CUSTID>ssdoe</CUSTID>
            <INSURED>
                <PREFIX>Mr.</PREFIX>
                <FNAME>Steven</FNAME>
                <MNAME>S</MNAME>
                <LNAME>Doe</LNAME>
                <SEX>M</SEX>
                <ADDRESS1>2727 Paces Ferry Road</ADDRESS1>
                <ADDRESS2>Apartment 900</ADDRESS2>
                <CITY>Atlanta</CITY>
                <STATE>GA</STATE>
                <ZIP>30339</ZIP>
                <BIRTHDATE>14-FEB-1970</BIRTHDATE>
                <INSSSAN>012345678</INSSSAN>
                <DAYPHONE>2148762789778</DAYPHONE>
                <NIGHTPHONE>2148974464</NIGHTPHONE>
                <BIRTHCITY>Pittsburg</BIRTHCITY>
                <BIRTHSTATE>PN</BIRTHSTATE>
                <DRIVERSTATE>GA</DRIVERSTATE>
                <DRIVERLICENSE>987987YIU</DRIVERLICENSE>
            </INSURED>
            <AGENT>
                <PREFIX>Mr.</PREFIX>
                <FNAME>John</FNAME>
                <LNAME>Doe</LNAME>
                <ADDRESS1>1100 Abernathy Road</ADDRESS1>
                <CITY>Atlanta</CITY>
                <STATE>GA</STATE>
                <ZIP>30328</ZIP>
                <EMAIL>johndoe@example.com</EMAIL>
                <PHONE>2148582200</PHONE>
                <AgentNo>R98798</AgentNo>
                <CustServPhone>8882637436</CustServPhone>
                <CustServOpenTime>8:00</CustServOpenTime>
                <CustServCloseTime>5:00</CustServCloseTime>
                <CustServTimeZone>eastern</CustServTimeZone>
            </AGENT>
            <POLICY_DATA>
                <PolicyValue>10000000</PolicyValue>
                <PolicyIssueDate>01032005</PolicyIssueDate>
                <PolicyEndDate>01032025</PolicyEndDate>
            <IssueState>GA</IssueState>
            <CostofInsurance>99200</CostofInsurance>
            <CostofInsuranceRate>992</CostofInsuranceRate>
            <CostofInsurance_Option>Level</CostofInsurance_Option>
            <Smoker>N</Smoker>
            <DeathBenefitType>Increasing</DeathBenefitType>
            <AnnualPremium>101900</AnnualPremium>
            <PremiumFrequency>Monthly</PremiumFrequency>
            <PremiumAmount>8492</PremiumAmount>
            <FlatExtra>0</FlatExtra>
            <AdminCharges>2700</AdminCharges>
            <MultipleExtra>0</MultipleExtra>
        </POLICY_DATA>
        <BENEFICIARY>
            <Name>Mary Doe</Name>
            <Relationship>Wife</Relationship>
        </BENEFICIARY>
        <BENEFICIARY>
            <Name>Anna Doe</Name>
            <Relationship>Daughter</Relationship>
        </BENEFICIARY>
    </Data>
</InterfaceRequest>
```

Example Trn_Fields INI Settings

The Trn_Fields control group in the FSISYS.INI or FSIUSER.INI file defines new GVM variable names for the subset of data fields in the example XML extract file that you want to reference and use for destination system metadata — this file creates new simple names for the fields italicized in the *Example XML Extract File* on page 81.

```
; XPATH to data elements in XML import file listed above,
; stores data in named GVMs
;
;  GVM Name   = XPATH to XML input file field for value
;  -----------  ----------------------------------------------
< Trn_Fields >
   Key1       = !/InterfaceRequest/Header/Key1
   Key2       = !/InterfaceRequest/Header/Key2
   KeyID      = !/InterfaceRequest/Header/KeyID
   TranCode   = !/InterfaceRequest/Header/TRANCODE
   RunDate    = !/InterfaceRequest/Header/Run_Date
   Product    = !/InterfaceRequest/Header/PRODUCT
   SecGroup   = !/InterfaceRequest/Header/SECGROUP
   DocType    = !/InterfaceRequest/Header/DOCTYPE
   Cabinet    = !/InterfaceRequest/Header/CABINET
   Author     = !/InterfaceRequest/Header/AUTHOR
   CurrUser   = !/InterfaceRequest/Header/CURRUSER
   CustID     = !/InterfaceRequest/Header/CUSTID
   PolNum     = !/InterfaceRequest/Data/POLICY_NUMBER
   InsFName   = !/InterfaceRequest/Data/INSURED/FNAME
   InsLName   = !/InterfaceRequest/Data/INSURED/LNAME
   InsAdd1    = !/InterfaceRequest/Data/INSURED/ADDRESS1
   InsAdd2    = !/InterfaceRequest/Data/INSURED/ADDRESS2
   InsCity    = !/InterfaceRequest/Data/INSURED/CITY
   InsState   = !/InterfaceRequest/Data/INSURED/STATE
   InsZIP     = !/InterfaceRequest/Data/INSURED/ZIP
   InsPhone   = !/InterfaceRequest/Data/INSURED/DAYPHONE
   InsDOB     = !/InterfaceRequest/Data/INSURED/BIRTHDATE
   WIPReason  = !/InterfaceRequest/Data/WIPREASON
   Index01    = !/InterfaceRequest/Data/AGENT/AgentNo
   Index02    = !/InterfaceRequest/Data/EFFDATE
   Index03    = !/InterfaceRequest/Data/EXPDATE
   Index04    = !/InterfaceRequest/Data/AGENT/AgentNo
   Index05    = !/InterfaceRequest/Data/EFFDATE
   Index06    = !/InterfaceRequest/Data/EXPDATE
   Index07    = !/InterfaceRequest/Data/AGENT/AgentNo
   Index08    = !/InterfaceRequest/Data/EFFDATE
   Index09    = !/InterfaceRequest/Data/EXPDATE
   Index10    = !/InterfaceRequest/Data/AGENT/AgentNo
   Index11    = !/InterfaceRequest/Data/EFFDATE
   Index12    = !/InterfaceRequest/Data/EXPDATE
   AgencyID   = !/InterfaceRequest/Data/AGENT/AgentNo
   EFFDate    = !/InterfaceRequest/Data/EFFDATE
   EXPDate    = !/InterfaceRequest/Data/EXPDATE
```

Example TRNDFDFL.DFD File

With the fields mapped into GVM variables, the attributes of each new GVM variable are described to Documaker using the Fields and Field:XXX control groups. The GVM variables defined in the Trn_Fields control group are shown in *italics*.

```
< Fields >
   FieldName = PKG_Offset
   FieldName = TRN_Offset
   FieldName = X_Offset
   FieldName = NA_Offset
   FieldName = POL_Offset
   FieldName = SentToManualBatch
   FieldName = KEY1
   FieldName = KEY2
   FieldName = KEYID
   FieldName = TRANCODE
   FieldName = RUNDATE
   FieldName = CURRUSER
   FieldName = AGENCYID
   FieldName = EFFDATE
   FieldName = EXPDATE
   FieldName = PRODUCT
   FieldName = SECGROUP
   FieldName = AUTHOR
   FieldName = CABINET
   FieldName = DOCTYPE
   FieldName = ONE
   FieldName = TWO
   FieldName = CUSTID
   FieldName = POLNUM
   FieldName = INSFNAME
   FieldName = INSLNAME
   FieldName = INSADD1
   FieldName = INSADD2
   FieldName = INSCITY
   FieldName = INSSTATE
   FieldName = INSZIP
   FieldName = INSPHONE
   FieldName = INSDOB
   FieldName = WIPREASON
   FieldName = INDEX01
   FieldName = INDEX02
   FieldName = INDEX03
   FieldName = INDEX04
   FieldName = INDEX05
   FieldName = INDEX06
   FieldName = INDEX07
   FieldName = INDEX08
   FieldName = INDEX09
   FieldName = INDEX10
   FieldName = INDEX11
   FieldName = INDEX12
< Field:PKG_Offset >
   INT_Type   = LONG
   EXT_Type   = CHAR_ARRAY_NO_NULL_TERM
   EXT_Length= 10
   Key        = No
   Required   = No
< Field:TRN_Offset >
   Int_Type   = LONG
   Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
   Ext_Length = 10
```

```
    Key        = No
    Required   = No
< Field:X_Offset >
    Int_Type   = LONG
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 10
    Key        = No
    Required   = No
< Field:NA_Offset >
    Int_Type   = LONG
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 10
    Key        = No
    Required   = No
< Field:POL_Offset >
    Int_Type   = LONG
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 10
    Key        = No
    Required   = No
< Field:SentToManualBatch >
    Int_Type   = CHAR_ARRAY
    Int_Length= 3
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 2
    Key        = No
    Required   = No
< Field:KEY1 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 101
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 100
    Key        = Yes
    Required   = Yes
< Field:KEY2 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 101
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 100
    Key        = Yes
    Required   = Yes
< Field:KEYID >
    Int_Type   = CHAR_ARRAY
    Int_Length= 101
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 100
    Key        = No
    Required   = No
< Field:TRANCODE >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:RUNDATE >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:CURRUSER >
    Int_Type   = CHAR_ARRAY
    Int_Length= 101
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 100
    Key       = No
    Required  = No
< Field:AGENCYID >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:EFFDATE >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:EXPDATE >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:PRODUCT >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:SECGROUP >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:AUTHOR >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:CABINET >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:DOCTYPE >
    Int_Type  = CHAR_ARRAY
    Int_Length= 31
    Ext_Type  = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key       = No
    Required  = No
< Field:ONE >
    Int_Type  = CHAR_ARRAY
    Int_Length= 11
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 10
    Key        = Yes
    Required   = Yes
< Field:TWO >
    Int_Type   = CHAR_ARRAY
    Int_Length= 11
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 10
    Key        = Yes
    Required   = Yes
< Field:CUSTID >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:POLNUM >
    Int_Type   = CHAR_ARRAY
    Int_Length= 101
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 100
    Key        = No
    Required   = No
< Field:INSFNAME >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INSLNAME >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INSADD1 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 101
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 100
    Key        = No
    Required   = No
< Field:INSADD2 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 101
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 100
    Key        = No
    Required   = No
< Field:INSCITY >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INSSTATE >
    Int_Type   = CHAR_ARRAY
    Int_Length= 4
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 3
    Key        = No
    Required   = No
< Field:INSZIP >
    Int_Type   = CHAR_ARRAY
    Int_Length= 12
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 11
    Key        = No
    Required   = No
< Field:INSPHONE >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INSDOB >
    Int_Type   = CHAR_ARRAY
    Int_Length = 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:WIPREASON >
    Int_Type   = CHAR_ARRAY
    Int_Length= 26
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 25
    Key        = No
    Required   = No
< Field:INDEX01 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX02 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX03 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX04 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX05 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX06 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX07 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX08 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX09 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX10 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX11 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
< Field:INDEX12 >
    Int_Type   = CHAR_ARRAY
    Int_Length= 31
    Ext_Type   = CHAR_ARRAY_NO_NULL_TERM
    Ext_Length= 30
    Key        = No
    Required   = No
```

EXAMPLE RCBDFDFL.DFD FILE

For this example, the RCBDFDFL.DFD file is identical to the TRNDFDFL.DFD file.

USING DAL TO OUTPUT TO A DATABASE TABLE

Here is an example of the DDL statements you could use to create a database table for use by Documaker Server and Documaker Connector. Documaker DAL writes to this table and Documaker Connector reads the records to determine which incoming documents to process.

Note Below is shown a typical example. For information on the minimum requirements, see *Minimum DAL Output Database Table* on page 42.

```
CREATE TABLE "ORACLE"."AOR" (
   "JOBID" VARCHAR2(50) NOT NULL,
   "TRANID" VARCHAR2(50) NOT NULL,
   "BATCHID" VARCHAR2(50) NOT NULL,
   "DOCID" VARCHAR2(50) NOT NULL,
   "NAME" VARCHAR2(30),
   "TYPE" VARCHAR2(30),
   "TITLE" VARCHAR2(255),
   "AUTHOR" VARCHAR2(50),
   "SECGROUP" VARCHAR2(30),
   "CABINET" VARCHAR2(30),
   "PFILE" VARCHAR2(255),
   "STATUSCD" INTEGER DEFAULT 0 NOT NULL,
   "STARTTIME" TIMESTAMP,
   "ENDTIME" TIMESTAMP,
   "RESULTDESC" VARCHAR2(2000),
   "RETENTION" TIMESTAMP,
   "CATEGORY" VARCHAR2(30),
   "KEY1" VARCHAR2(100),
   "KEY2" VARCHAR2(100),
   "KEYID" VARCHAR2(100),
   "TRANCODE" VARCHAR2(30),
   "RUNDATE" TIMESTAMP,
   "CURRUSER" VARCHAR2(30),
   "AGENCYID" VARCHAR2(30),
   "EFFDATE" TIMESTAMP,
   "EXPDATE" TIMESTAMP,
   "CUSTID" VARCHAR2(30),
   "POLNUM" VARCHAR2(100),
   "INSFNAME" VARCHAR2(30),
   "INSLNAME" VARCHAR2(30),
   "INSADD1" VARCHAR2(30),
   "INSADD2" VARCHAR2(30),
   "INSCITY" VARCHAR2(30),
   "INSSTATE" VARCHAR2(5),
   "INSZIP" VARCHAR2(30),
   "INSPHONE" VARCHAR2(30),
   "INSDOB" DATE,
   "INDEX01" VARCHAR2(30), "INDEX02" VARCHAR2(30), "INDEX03"
VARCHAR2(30),
   "INDEX04" VARCHAR2(30), "INDEX05" VARCHAR2(30), "INDEX06"
VARCHAR2(30),
   "INDEX07" VARCHAR2(30), "INDEX08" VARCHAR2(30), "INDEX09"
VARCHAR2(30),
   "INDEX10" VARCHAR2(30), "INDEX11" VARCHAR2(30), "INDEX12"
VARCHAR2(30),
   PRIMARY KEY ("JOBID", "TRANID", "BATCHID", "DOCID") VALIDATE
);

CREATE INDEX "ORACLE"."AORIDX1" ON "ORACLE"."AOR" ("DOCID");
```

EXAMPLE DFD FILE (AOR.DFD)

This DFD (Data Format Definition) file describes the database table schema to Documaker. In this example, with a database table called *AOR*, this file is called *AOR.DFD* and is used to define the interface between Documaker and the data table AOR in the Oracle database used for storage of the extract data.

The DFD file is used in the DAL script where you will see references to AOR in the DBAdd, DBOpen, and DBPrepare function calls. This file is loaded via DBOpen so Documaker knows what columns are available. Documaker also uses that information to insert the row in the AOR table.

In the example, this file is in the deflib subdirectory, per the call in the DAL script:

```
DBOPEN(AOR_TableName, "ODBC", ".\deflib\aor.dfd", "READ&WRITE&CREATE_IF_NEW")
```

Here is an example of the AOR.DFD file:

```
< Fields >
   FieldName = JOBID
   FieldName = TRANID
   FieldName = BATCHID
   FieldName = DOCID
   FieldName = NAME
   FieldName = TYPE
   FieldName = TITLE
   FieldName = AUTHOR
   FieldName = SECGROUP
   FieldName = PFILE
   FieldName = CATEGORY
   FieldName = CABINET
   FieldName = STATUSCD
   FieldName = KEY1
   FieldName = KEY2
   FieldName = KEYID
   FieldName = TRANCODE
   FieldName = RUNDATE
   FieldName = CURRUSER
   FieldName = AGENCYID
   FieldName = EFFDATE
   FieldName = EXPDATE
   FieldName = INSFNAME
   FieldName = INSLNAME
   FieldName = INSADD1
   FieldName = INSADD2
   FieldName = INSCITY
   FieldName = INSSTATE
   FieldName = INSZIP
   FieldName = INSPHONE
   FieldName = INSDOB
   FieldName = INDEX01
   FieldName = INDEX02
   FieldName = INDEX03
   FieldName = INDEX04
   FieldName = INDEX05
   FieldName = INDEX06
   FieldName = INDEX07
   FieldName = INDEX08
   FieldName = INDEX09
   FieldName = INDEX10
   FieldName = INDEX11
   FieldName = INDEX12
< Field:JOBID >
   Int_Type = CHAR_ARRAY
```

```
    Int_Length = 47
    Ext_Type = CHAR_ARRAY
    Ext_Length = 47
    Key = Y
    Required = Y
< Field:TRANID >
    Int_Type = CHAR_ARRAY
    Int_Length = 47
    Ext_Type = CHAR_ARRAY
    Ext_Length = 47
    Key = Y
    Required = Y
< Field:BATCHID >
    Int_Type = CHAR_ARRAY
    Int_Length = 47
    Ext_Type = CHAR_ARRAY
    Ext_Length = 47
    Key = Y
    Required = Y
< Field:DOCID >
    Int_Type = CHAR_ARRAY
    Int_Length = 47
    Ext_Type = CHAR_ARRAY
    Ext_Length = 47
    Key = Y
    Required = Y
< Field:NAME >
    Int_Type = CHAR_ARRAY
    Int_Length = 47
    Ext_Type = CHAR_ARRAY
    Ext_Length = 47
    Key = N
    Required = Y
< Field:TYPE >
    Int_Type = CHAR_ARRAY
    Int_Length = 10
    Ext_Type = CHAR_ARRAY
    Ext_Length = 10
    Key = N
    Required = Y
< Field:TITLE >
    Int_Type = CHAR_ARRAY
    Int_Length = 30
    Ext_Type = CHAR_ARRAY
    Ext_Length = 30
    Key = N
    Required = Y
< Field:AUTHOR >
    Int_Type = CHAR_ARRAY
    Int_Length = 30
    Ext_Type = CHAR_ARRAY
    Ext_Length = 30
    Key = N
    Required = Y
< Field:SECGROUP >
    Int_Type = CHAR_ARRAY
    Int_Length = 30
    Ext_Type = CHAR_ARRAY
    Ext_Length = 30
    Key = N
    Required = Y
< Field:PFILE >
    Int_Type = CHAR_ARRAY
    Int_Length = 255
```

```
   Ext_Type = CHAR_ARRAY
   Ext_Length = 255
   Key = N
   Required = Y
< Field:CATEGORY >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = N
   Required = N
< Field:CABINET >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = N
   Required = N
< Field:STATUSCD >
   Int_Type = LONG
   Int_Length = 1
   Ext_Type = LONG
   Ext_Length = 1
   Key = N
   Required = Y
< Field:KEY1 >
   Int_Type = CHAR_ARRAY
   Int_Length = 100
   Ext_Type = CHAR_ARRAY
   Ext_Length = 100
   Key = N
   Required = N
< Field:KEY2 >
   Int_Type = CHAR_ARRAY
   Int_Length = 100
   Ext_Type = CHAR_ARRAY
   Ext_Length = 100
   Key = N
   Required = N
< Field:KEYID >
   Int_Type = CHAR_ARRAY
   Int_Length = 100
   Ext_Type = CHAR_ARRAY
   Ext_Length = 100
   Key = N
   Required = N
< Field:TRANCODE >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = N
   Required = N
< Field:RUNDATE >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = N
   Required = N
< Field:CURRUSER >
   Int_Type = CHAR_ARRAY
   Int_Length = 100
   Ext_Type = CHAR_ARRAY
```

```
       Ext_Length = 100
       Key = N
       Required = N
   < Field:AGENCYID>
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
   < Field:EFFDATE >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
   < Field:EXPDATE >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
   < Field:CUSTID >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
   < Field:POLNUM >
       Int_Type = CHAR_ARRAY
       Int_Length = 100
       Ext_Type = CHAR_ARRAY
       Ext_Length = 100
       Key = N
       Required = N
   < Field:INSFNAME >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
   < Field:INSLNAME >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
   < Field:INSADD1 >
       Int_Type = CHAR_ARRAY
       Int_Length = 100
       Ext_Type = CHAR_ARRAY
       Ext_Length = 100
       Key = N
       Required = N
   < Field:INSADD2 >
       Int_Type = CHAR_ARRAY
       Int_Length = 100
       Ext_Type = CHAR_ARRAY
       Ext_Length = 100
       Key = N
       Required = N
    < Field:INSCITY >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
    < Field:INSSTATE >
       Int_Type = CHAR_ARRAY
       Int_Length = 3
       Ext_Type = CHAR_ARRAY
       Ext_Length = 3
       Key = N
       Required = N
    < Field:INSZIP >
       Int_Type = CHAR_ARRAY
       Int_Length = 11
       Ext_Type = CHAR_ARRAY
       Ext_Length = 11
       Key = N
       Required = N
    < Field:INSPHONE >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
    < Field:INSDOB >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
    < Field:WIPREASON >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
    < Field:INDEX01 >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
    < Field:INDEX02 >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
    < Field:INDEX03 >
       Int_Type = CHAR_ARRAY
       Int_Length = 30
       Ext_Type = CHAR_ARRAY
       Ext_Length = 30
       Key = N
       Required = N
```

```
< Field:INDEX04 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = N
   Required = N
< Field:INDEX05 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Field:INDEX06 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Field:INDEX07 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Field:INDEX08 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Field:INDEX09 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Field:INDEX10 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Field:INDEX11 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Field:INDEX12 >
   Int_Type = CHAR_ARRAY
   Int_Length = 30
   Ext_Type = CHAR_ARRAY
   Ext_Length = 30
   Key = No
   Required = No
< Keys >
```

```
   KeyName = BATCH
   KeyName = DOCID
< Key:BATCH >
   Expression = JOBID+TRANID+BATCHID
   FieldList = JOBID,TRANID,BATCHID
< Key:DOCID >
   Expression = DOCID
   FieldList = DOCID
```

Example DAL Scripts

These scripts are referenced in the DAL configuration of the Documaker INI file. This configuration was described earlier, but is presented here for reference:

```
; Enable the Banner and Transaction DAL Scripting
< Printer >
; Must generally enable banner processing for it to work.
   EnableBatchBanner = Yes
   EnableTransBanner = Yes
   PrtType = PDF
< Batch6 >
   EnableBatchBanner       = Yes
   EnableTransBanner       = Yes
   BatchBannerBeginScript  = AOR_PREB
   BatchBannerEndScript    = AOR_POSTB
   TransBannerBeginScript  = AOR_PRET
   TransBannerEndScript    = AOR_POSTT
   Printer                 = Printer6
```

In the example presented earlier, they are called during processing of the Batch6 grouping. The code for the routines is shown below, with the main entry points listed above shown below in the order they are called. The *TransBanner* routines are called repeatedly for each transaction that is part of the batch, before the final call to BatchBannerEndScript.

```
   BatchBannerBeginScript     = AOR_PREB
      TransBannerBeginScript  = AOR_PRET
      TransBannerEndScript    = AOR_POSTT
   BatchBannerEndScript       = AOR_POSTB
```

BatchBannerBeginScript = AOR_PREB

```
BEGINSUB AOR_PREB
* ------------------------------------------------------
*   Begin batch
*   Clear variables once per recipient batch
* ------------------------------------------------------
   #AOR_Debug=GETINIBOOL(,PRINTERID(),"AORDebug")

   IF #AOR_Debug
      RPLogMsg(NL() & "  ** AOR_PREB:" & NL() )
   END
   AOR_RecipBatch   = AOR_RecipBatch
   #AOR_BatchCount  = #AOR_BatchCount
   #AOR_Processed   = #AOR_Processed
   #AOR_Count       = #AOR_Count
   AOR_TableName    = AOR_TableName
   #AOR_Init        = #AOR_Init
   IF AOR_RecipBatch != RECIPBATCH()
      PUTINIBOOL(,"RunMode","CheckNextRecip",0)
      #AOR_PerBatch   =
GETINISTRING(,PRINTERID(),"AORFilesPerBatch","999")
      AOR_RecipBATCH   = RECIPBATCH()
      #AOR_SubBatch    = 0
      #AOR_Count       = 0
      #AOR_BatchCount = 0
   END
   AOR_BatchID = RECIPBATCH()
   IF #AOR_Init = 0
      AOR_JobID = UNIQUESTRING()
      AOR_TableName = GETINISTRING(,PRINTERID(),"AORTable","AOR")
```

```
       DBOPEN(AOR_TableName,"ODBC",".\deflib\aor.dfd",
"READ&WRITE&CREATE_IF_NEW")
       DBPREPVARS(AOR_TableName,"AORTABLERecord")
   END
   #AOR_Init = 1
   #AOR_DoEOB = 1
ENDSUB
```

TransBannerBeginScript = AOR_PRET

```
BEGINSUB AOR_PRET
* -------------------------------------------------------
*   Begin Transaction
*   Set up new file name for recipient batch output file
* -------------------------------------------------------
IF #AOR_Debug
       RPLogMsg(NL() & "  ** AOR_PRET:" & NL() )
   END
   AOR_BatchID     = AOR_BatchID
   AOR_BatchDir    = AOR_BatchDir
   #AOR_Batch      = #AOR_Batch
   #AOR_Count      = #AOR_Count
   #AOR_PerBatch   = #AOR_PerBatch
   #AOR_Processed = #AOR_Processed
   AOR_TransID = GVM("KEY1") & "-" & GVM("KEY2") & "-" & \
       GVM("KEYID") & "-" & GVM("TRANCODE")
   #AOR_Count += 1
   IF (#AOR_Count > #AOR_PerBatch)
       #AOR_Count -=1
       CALL("AOR_EOB")
       #AOR_Count = 1
   END
   TranFile = CALL("AOR_NEWFILE")
   #AOR_Exists  = PATHEXIST(AOR_BatchDir)
   IF #AOR_Exists = 0
       PATHCREATE(AOR_BatchDir)
       #AOR_Exists  = PATHEXIST(AOR_BatchDir)
       IF #AOR_Exists = 0
           RPErrorMsg(NL() & "** AOR batch directory " & \
               AOR_Batchdir & "does not exist!")
       END
   END
   #AOR_DoEOB = 0
   SETDEVICENAME(TranFile)
   BREAKBATCH()
ENDSUB
```

TransBannerEndScript = AOR_POSTT

```
BEGINSUB AOR_POSTT
* -------------------------------------------------------
*   End Transaction
*   Insert new table row with metadata for archive
*   from GVM variables and reference to uniquely named
*   recipient print stream output.
* -------------------------------------------------------

    IF #AOR_Debug
       RPLogMsg(NL() & "  ** AOR_POSTT:" & NL() )
    END
    #AOR_PerBatch = #AOR_PerBatch
    AOR_BatchDir = AOR_BatchDir
    #AOR_Count = #AOR_Count
    #AOR_Debug = GETINIBOOL(,PRINTERID(),"AORDebug")
    AOR_TableName = AOR_TableName
    AOR_BatchID = AOR_BatchID
    AOR_TransID = AOR_TransID
    AOR_JobID   = AOR_JobID
*
*       Change to match variables defined in "rcbdfdfl.dfd"
*       as needed for the implementation
*

* Documaker Connector job processing fields example for UCM
    AORTABLERecord.JOBID    = AOR_JobID
    AORTABLERecord.BATCHID  = AOR_BatchID
    AORTABLERecord.TRANID   = AOR_TransID
    AORTABLERecord.DOCID    = AOR_FName
* TITLE required field by UCM 30 characters max, shows up in search 
results
    AORTABLERecord.TITLE    = AOR_TransID
* ContentID/Name assigment
    AORTABLERecord.STATUSCD = 0
* Documaker UCMImporter required field for specifying full name of
* file to import
    AORTABLERecord.PFILE    = DEVICENAME()
* UCM required field has to exist in UCM pick list for types or
* will fail to import
    IF HAVEGVM("DOCTYPE")
        AORTABLERecord.TYPE      = GVM("DOCTYPE")
    END
* UCM required field
    IF HAVEGVM("AUTHOR")
        AORTABLERecord.AUTHOR    = GVM("AUTHOR")
    END
* UCM required field has to exist in UCM pick list for security groups 
or will
* fail to import
    IF HAVEGVM("SECGROUP")
        AORTABLERecord.SECGROUP  = GVM("SECGROUP")
    END
* UCM mapped custom meta-data, if doesn't exist as same exact name in 
UCM custom
* fields it will not map but will not error. If UCM custom field was 
set as
* required and no data is mapped UCM will fail transaction.
* Truncates by default to the max
* length of UCM data type.
    IF HAVEGVM("CABINET")
        AORTABLERecord.CABINET   = GVM("CABINET")
    END
    IF HAVEGVM("KEY1")
        AORTABLERecord.KEY1      = GVM("KEY1")
    END
    IF HAVEGVM("KEY2")
        AORTABLERecord.KEY2      = GVM("KEY2")
    END
    IF HAVEGVM("KEYID")
        AORTABLERecord.KEYID     = GVM("KEYID")
    END
    IF HAVEGVM("TRANCODE")
        AORTABLERecord.TRANCODE  = GVM("TRANCODE")
    END
    IF HAVEGVM("RUNDATE")
        AORTABLERecord.RUNDATE  = GVM("RUNDATE")
    END
    IF HAVEGVM("CURRUSER")
        AORTABLERecord.CURRUSER  = GVM("CURRUSER")
    END
    IF HAVEGVM("AGENCYID")
        AORTABLERecord.AGENCYID  = GVM("AGENCYID")
    END
    IF HAVEGVM("EFFDATE")
        AORTABLERecord.EFFDATE  = GVM("EFFDATE")
        AORTABLERecord.TITLE    = AOR_TransID & "-" & GVM("EFFDATE")
    END
    IF HAVEGVM("EXPDATE")
        AORTABLERecord.EXPDATE  = GVM("EXPDATE")
    END
    IF HAVEGVM("CUSTID")
        AORTABLERecord.CUSTID  = GVM("CUSTID")
    END
    IF HAVEGVM("POLNUM")
        AORTABLERecord.POLNUM  = GVM("POLNUM")
    END
    IF HAVEGVM("INSFNAME")
        AORTABLERecord.INSFNAME  = GVM("INSFNAME")
    END
    IF HAVEGVM("INSLNAME")
        AORTABLERecord.INSLNAME  = GVM("INSLNAME")
    END
    IF HAVEGVM("INSADD1")
        AORTABLERecord.INSADD1  = GVM("INSADD1")
    END
    IF HAVEGVM("INSADD2")
        AORTABLERecord.INSADD2  = GVM("INSADD2")
    END
    IF HAVEGVM("INSCITY")
        AORTABLERecord.INSCITY  = GVM("INSCITY")
    END
    IF HAVEGVM("INSSTATE")
        AORTABLERecord.INSSTATE  = GVM("INSSTATE")
    END
    IF HAVEGVM("INSZIP")
        AORTABLERecord.INSZIP  = GVM("INSZIP")
    END
    IF HAVEGVM("INSPHONE")
        AORTABLERecord.INSPHONE  = GVM("INSPHONE")
    END
    IF HAVEGVM("INSDOB")
        AORTABLERecord.INSDOB  = GVM("INSDOB")
    END
    IF HAVEGVM("INDEX01")
        AORTABLERecord.INDEX01  = GVM("INDEX01")
    END
    IF HAVEGVM("INDEX02")
        AORTABLERecord.INDEX02  = GVM("INDEX02")
    END
    IF HAVEGVM("INDEX03")
        AORTABLERecord.INDEX03  = GVM("INDEX03")
    END
    IF HAVEGVM("INDEX04")
        AORTABLERecord.INDEX04  = GVM("INDEX04")
    END
    IF HAVEGVM("INDEX05")
        AORTABLERecord.INDEX05  = GVM("INDEX05")
    END
    IF HAVEGVM("INDEX06")
        AORTABLERecord.INDEX06  = GVM("INDEX06")
    END
    IF HAVEGVM("INDEX07")
```

```
        AORTABLERecord.INDEX07   = GVM("INDEX07")
    END
    IF HAVEGVM("INDEX08")
        AORTABLERecord.INDEX08   = GVM("INDEX08")
    END
    IF HAVEGVM("INDEX09")
        AORTABLERecord.INDEX09   = GVM("INDEX09")
    END
    IF HAVEGVM("INDEX10")
        AORTABLERecord.INDEX10   = GVM("INDEX10")
    END
    IF HAVEGVM("INDEX11")
        AORTABLERecord.INDEX11   = GVM("INDEX11")
    END
    IF HAVEGVM("INDEX12")
        AORTABLERecord.INDEX12   = GVM("INDEX12")
    END
    #Rtn = DBADD(AOR_TableName,"AORTABLERecord")
ENDSUB
```

BatchBannerEndScript = AOR_POSTB

```
BEGINSUB AOR_POSTB
* ------------------------------------------------------
*   End Batch
*   Any necessary clean-up.
* ------------------------------------------------------

    IF #AOR_Debug
       RPLogMsg(NL() & "  ** AOR_POSTB:" & NL() )
    END

    #AOR_Debug = GETINIBOOL(,PRINTERID(),"AORDebug")
    #AOR_Processed = #AOR_Processed
    #AOR_Count = #AOR_Count
    #AOR_PerBatch = #AOR_PerBatch
    AOR_BatchDir = AOR_BatchDir
    AOR_TableName = AOR_TableName

    IF (#AOR_DoEOB = 1 AND #AOR_Count > 0)
        DBCLOSE(AOR_TableName)
        CALL("AOR_EOB")
        #AOR_Count = 0
    END
ENDSUB
```

Internal Routine: AOR_NEWFILE

```
BEGINSUB AOR_NEWFILE
* ------------------------------------------------------
*   Create a new output file name.
*   Called by Pre-Transaction DAL script.
* ------------------------------------------------------
    IF #AOR_Debug
       RPLogMsg(NL() & "  ** AOR_NEWFILE:" & NL() )
    END
    AOR_Ext = GETINISTRING(,PRINTERID(), "AORExt", ".pdf")
    AOR_BatchDir = AOR_BatchDir
    CALL("AOR_NEWPATH")
    AOR_Drive = FILEDRIVE( AOR_BatchDir )
    AOR_Path = FILEPATH( AOR_BatchDir )
    AOR_Last = FILENAME( AOR_BatchDir )
    AOR_FName = UNIQUESTRING()
    AOR_NewFName=FULLFILENAME( AOR_Drive,AOR_Path &
AOR_Last,AOR_FName,AOR_Ext)
    RETURN( AOR_NewFName )
ENDSUB
```

Internal Routine: AOR_NEWPATH

```
BEGINSUB AOR_NEWPATH
* ------------------------------------------------------
*   Create a new output folder.
*   Called by AOR_NEWFILE DAL script.
* ------------------------------------------------------
    IF #AOR_Debug
       RPLogMsg(NL() & "  ** AOR_NEWPATH:" & NL() )
    END
    #AOR_BatchCount = #AOR_BatchCount
    AOR_BatchID = AOR_BatchID
    AOR_BatchDir = AOR_BatchDir
    #AOR_Count = #AOR_Count
    AORPath = GETINISTRING(,PRINTERID(),"AORPath")
    AOR_Drive    = FILEDRIVE( AORPath )
    AOR_Path     = FILEPATH( AORPath )
    AOR_Last     = FILENAME( AORPath )
    AOR_Rootdir  = FULLFILENAME( AOR_Drive, AOR_Path, AOR_Last, "")
    IF #AOR_Count = 1
        #AOR_BatchCount += 1
        AOR_Drive    = FILEDRIVE(AOR_RootDir)
        AOR_Path     = FILEPATH(AOR_RootDir)
        AOR_Last     = FILENAME(AOR_RootDir)
        AOR_BatchDir = FULLFILENAME(AOR_Drive,AOR_Path &
AOR_Last,AOR_JobID,"")
        AOR_Drive    = FILEDRIVE(AOR_BatchDir)
        AOR_Path     = FILEPATH(AOR_BatchDir)
        AOR_Last     = FILENAME(AOR_BatchDir)
        AOR_BatchDir =
FULLFILENAME(AOR_Drive,AOR_Path&AOR_Last,AOR_BatchID&"x"&#AOR_BatchC
ount,"")
    END
ENDSUB
```

Internal Routine: AOR_EOB

```
BEGINSUB AOR_EOB
* ------------------------------------------------------
*   Create transact.dat file listing the contents of
*   each batch folder created with count of output files,
*   maximum output files per batch folder, statistical
*   information. Existence of this file is indicates a
*   batch folder has been completed and is used by the
*   Documaker Connector source for housekeeping functions.
*   Called by Post Batch DAL script.
* ------------------------------------------------------
    IF #AOR_Debug
       RPLogMsg(NL() & "  ** AOR_EOB:" & NL() )
    END
    #AOR_PerBatch = #AOR_PerBatch
    #AOR_Count = #AOR_Count
    AOR_BatchDir = AOR_BatchDir
    AORRootDir = AORRootDir
    IF #AOR_DoEOB = 1
       AOR_LogFile = FULLFILENAME(,AOR_BatchDir,"transact",".dat")
       DBOPEN(AOR_LogFile,"ASCII",".\deflib\aort.dfd", \
                  "READ&WRITE&TRUNCATE&CREATE_IF_NEW")
       DBPREPVARS(AOR_LogFile,"AOREOTRecord")

       AOREOTRecord.Record = FILENAME(AOR_BatchDir) & " " & \
                             #AOR_PerBatch & " " & \
                             #AOR_Count
       DBADD(AOR_LogFile,"AOREOTRecord")
       DBCLOSE(AOR_LogFile)
    END
ENDSUB
```