



Opsware® System DCML Exchange Tool 2.0 Reference Guide

Corporate Headquarters

599 North Mathilda Avenue Sunnyvale, California 94085 U.S.A.

T + 1 408.744.7300 F +1 408.744.7383 www.opsware.com

Copyright © 2000-2005 Opsware Inc. All Rights Reserved.

Opsware Inc. Unpublished Confidential Information. NOT for Redistribution. All Rights Reserved.

Opsware is protected by U.S. Patent Nos. 6,658,426, 6,751,702, 6,816,897, 6,763,361 and patents pending

Opsware, Opsware Command Center, Model Repository, Data Access Engine, Web Services Data Access Engine, Software Repository, Command Engine, Opsware Agent, Multimaster Replication Engine, and Code Deployment & Rollback are trademarks and service marks of Opsware Inc. All other marks mentioned in this document are the property of their respective owners.

Additional proprietary information about third party and open source materials can be found at <http://www.opsware.com/support/opensourcedoc.pdf>.

Table of Contents

Table of Contents	i
Preface	iii
About This Guide	iv
Contents of this Guide	iv
Conventions in this Guide	v
Icons in this Guide	v
Chapter 1: Installing and Configuring DCML Exchange Tool1	
Installing and Configuring DCML Exchange Tool (DET)	1
Creating A Target Core Configuration File	2
Distribution Directory	6
Chapter 2: Using the DCML Exchange Tool	9
Introduction to the DCML Exchange Tool (DET)	9
DET Relationship to DCML	10
Exporting Content	10
Export Filters	11

Example Export Filter File	12
Application Export Filter	13
Patch Export Filter	15
OS Export Filter.....	17
Service Level Export Filter	19
Template Export Filter	21
Custom Extension Export Filter	23
Distributed Script Export Filter.....	24
Application Configuration Export Filter.....	25
Application Configuration Template Export Filter.....	26
Server Compliance Criteria Export Filter	27
Importing Content	29
Policy on Importing Content Types	29
Additional Considerations When Importing Customers.....	34
Content Directory	35
Appendix A: DCML Exchange Tool Command Line	37
About DCML Exchange Tool (DET) Command Line	37
Commands and Options	38
Export Command	38
Import Command	39
Show Export Status Command	40
Configuration File Command	40
Show Version Command.....	41
Show Help Command	41
DET Permissions Command	41

Preface

Welcome to the Opsware DCML Exchange Tool (DET). This tool enables you to export almost all server management content from any Opsware core – standalone or multimaster mesh – and import it into any other Opsware core. Opsware can also provide pre-packaged server management content appropriate for new installations that can be imported into a core after initial setup. See your Opsware Representative for information on obtaining this content.

Content that can be exported includes:

- Applications
- Patches
- Operating Systems
- Templates
- Service Levels
- Distributed Scripts
- Custom Extensions
- Application Configurations
- Application Configuration Templates
- Server Compliance Criteria

In addition, information associated with the content is also exported, including:

- Customers
- Packages
- MRLs
- Install Hooks
- Units
- Install Order Relationships

- Custom Attributes
- Software Lists
- Configuration Tracking Policies
- Unit Scripts

About This Guide

This guide describes how to use the DCML Exchange Tool and provides information about how to export and import content, how to create export filters, how to specify handling of duplicates encountered during the import process, and complete information about command line options and arguments.

This guide is intended for system administrators who are responsible for specifying the content used on Opsware-managed servers.

Contents of this Guide

This guide contains the following chapters and appendices:

Chapter 1: Installing and Configuring the DCML Exchange Tool – provides information about the installation and configuration of the DCML Exchange Tool, including what files to download and where to find them, how to set the environment variable, how to obtain needed certificates, how to create a configuration file and what properties to use, and example configuration files.

Chapter 2: Using the DCML Exchange Tool – provides information about how to export content, what kind of content can be exported, commands and arguments for exports, how to create filters for exporting content and example filters, how to import content, policies for importing each content type, information about how each content type handles importing duplicates, and a description of the content directories

Appendix A: DCML Exchange Tool Command Line – provides information about command line commands and their associated arguments.





Conventions in this Guide

This guide uses the following typographical and formatting conventions.

NOTATION	DESCRIPTION
<code>Courier</code>	Identifies text of displayed messages and other output from Opsware programs or tools.
<code>Courier Bold</code>	Identifies user-entered text (commands or information).
<i><code>Courier Italics</code></i>	Identifies variable user-entered text on the command line or within example files.

Icons in this Guide

This guide uses the following iconographic conventions.

ICON	DESCRIPTION
	This icon is a note. It identifies especially important concepts that warrant added emphasis.
	This icon is a requirement. It identifies a task that must be performed before an action under discussion can be performed.
	This icon is a tip. It identifies information that can help simplify or clarify tasks.
	This icon is a warning. It is used to identify significant information that must be read before proceeding.

Chapter 1: Installing and Configuring DCML Exchange Tool

IN THIS CHAPTER

This chapter contains the following topic:

- Installing and Configuring DCML Exchange Tool (DET)

Installing and Configuring DCML Exchange Tool (DET)

The following instructions detail how to install and configure the DCML Exchange Tool (DET).



The DCML Exchange Tool (DET) can be run on any UNIX computer, though not necessarily a managed server. (Although DET is not supported on the Windows platform, it does support import and export of Windows content.)

- 1** Log on to an Opsware-managed server as the root user.
- 2** If you do not already have them, download JRE 1.4.x or JDK 1.4.x from www.sun.com, and install the program on the server where you have logged in.
- 3** Download the `cbt-<version>.zip` file from download.opsware.com and unzip the distribution on the server where you have logged in. You will need a login ID and password for the download site; ask your Opsware administrator if you do not have a login ID and password.
- 4** Set your `JAVA_HOME` environment variable to point to a Java 1.4.x installation. For example, in `csh` you would issue the following command:

```
% setenv JAVA_HOME <j2re 1.4.x installation>
```
- 5** Optionally, you can set the `PATH` environment variable to include the DET install directory: `<cbt-install_dir>/bin`.

6 Perform the following steps for each core that DET will be importing into or exporting from.

1. Log in to the Opsware Command Center, create a new user, and then add that user to the advanced users group.
2. Grant that user permission to export and import by executing the `cbtperm` command on the Opsware-managed server where the import and export will be performed:

```
% cd <cbt_install_dir>
% bin/cbtperm -u <user> -a <Opsware administrator> -p
<spike.port> -s <spike.host> -c <ssl.trustCerts> -k
<ssl.keyPairs>
Enter password for <Opsware administrator>: *****
- <user> now has permission to use the DCML Exchange Tool.
%
```
3. Obtain a copy of the `opsware-ca.crt` trust certificate from `/var/1c/crypto/twist/opsware-ca.crt` and save it in a location DET can access. This step is optional if you are running DET from the server where the Opsware Command Center is installed.
4. Obtain a copy of the `spog.pkcs8` client certificate from `/var/1c/crypto/twist/spog.pkcs8` and save it in a location DET can access. This step is optional if you are running DET from the server where the Opsware Command Center is installed.
5. Obtain the twist username and password - this is set during the twist install and the Opsware administrator should have this information.
6. Create a target core configuration file that contains the location and identity information required to access the Opsware core components.

Creating A Target Core Configuration File

Creating a target core configuration file simplifies the use of DET. This configuration file contains Opsware component access information that would otherwise need to be given on the DET command-line. The core configuration file is a key=value pair text file.

The following table contains all possible DET configuration-related properties. These properties can be either given on the DET command-line or specified in a configuration file.

The default configuration property values listed in Table 1-1 assume that you are running DET on an Opsware core running the Opsware Command Center. (It is for this reason that the `.host` properties shows a `localhost` value.) Also, `twist.certpaths`, `ssl.trustcerts`, and `ssl.keypairs` assume paths on an Opsware Command Center server.



If a configuration-related property is not specifically mentioned in the Core Configuration file, the default value shown in the Configuration Properties table below will be used.

Table 1-1: Configuration Properties

PROPERTY NAME	DEFAULT VALUE	DESCRIPTION
<code>cbt.numthreads</code>	1	Number of concurrent threads used for export. For exporting content, you can specify as many threads as you wish. However, for importing content, DET 2.0 supports only one thread.
<code>spike.enabled</code>	<code>true</code>	Use Spike for authentication and authorization on all XML-RPC-based servers.
<code>spike.host</code>	<code>way</code>	Spike's host name or IP.
<code>spike.path</code>	<code>wayrpc.py</code>	Spike's base URL path.
<code>spike.password</code>	<no default>	User password for Spike authentication. This is an OCC user's password and is set during the installation of the core. Contact your Opsware Administrator (or the person who installed the core) for this information.

Table 1-1: Configuration Properties

PROPERTY NAME	DEFAULT VALUE	DESCRIPTION
<code>spike.port</code>	1018	Spike's listener port.
<code>spike.protocol</code>	https	Spike's listener protocol. This is typically HTTPS.
<code>spike.username</code>	admin	User name for Spike authentication. This is the user who was granted permissions by the cbtperm tool. Default is the Opsware administrator.
<code>spin.host</code>	spin	Data Access Engine's host name or IP.
<code>spin.path</code>	spinrpc.py	Data Access Engine's base URL path.
<code>spin.port</code>	1004	Data Access Engine's listener port.
<code>spin.protocol</code>	http	Data Access Engine's listener protocol. HTTP if the DET is on the same server as the Opsware Command Center and is running a cleartext spin in a multi-server core or HTTPS for any other configuration.
<code>ssl.keyPairs</code>	/var/1c/crypto/ twist/spog.pkcs8	Comma-separated list of client certificates used to communicate with XML-RPC-based servers.
<code>ssl.trustCerts</code>	/var/1c/crypto/ twist/opsware-ca.crt	Comma-separated list of trust certificate files used to communicate with XML-RPC-based servers.

Table 1-1: Configuration Properties

PROPERTY NAME	DEFAULT VALUE	DESCRIPTION
<code>ssl.useHttpClient</code>	<code>true</code>	Use the HttpClient library instead of JDK's built-in HTTP client.
<code>twist.certpaths</code>	<code>/var/1c/crypto/ twist/opsware-ca.crt</code>	Comma-separated list of trust certificates used to communicate with the Web Services Data Access Engine.
<code>twist.host</code>	<code>localhost</code>	Web Services Data Access Engine's host name or IP.
<code>twist.password</code>	<code><no default></code>	Web Services Data Access Engine's password. This password is set during the installation of the core. Contact your Opsware Administrator (or the person who installed the core) for this information.
<code>twist.port</code>	<code>1032</code>	Web Services Data Access Engine's listening port.
<code>twist.protocol</code>	<code>t3s</code>	Web Services Data Access Engine's protocol. This should be <code>t3</code> or <code>t3s</code> .
<code>twist.username</code>	<code>detuser</code>	Web Services Data Access Engine's username (for example, "detuser").
<code>way.host</code>	<code>way</code>	Command Engine's host name or IP.
<code>way.path</code>	<code>wayrpc.py</code>	Command Engine's base URL path.

Table 1-1: Configuration Properties

PROPERTY NAME	DEFAULT VALUE	DESCRIPTION
way.port	1018	Command Engine's listener port.
way.protocol	https	Command Engine's listener protocol. This is typically HTTPS.
word.host	word	Software Repository's host name or IP.
word.path	wordrpc.py	Software Repository's base URL path.
word.port	1003	Software Repository's listener port.
word.protocol	https	Software Repository's listener protocol. This is HTTPS.

The following is an example of a target core configuration file that contains only essential core configuration information.

```
twist.host=twist.c07.dev.opsware.com
twist.port=1032
twist.protocol=t3s
twist.username=<detuser>
twist.password=<twist_password>
twist.certPaths=<absolute path to opsware-ca.crt>

spike.username=<OCC_user>
spike.password=<OCC_user_password>
spike.host=way.c07.dev.opsware.com
way.host=way.c07.dev.opsware.com
spin.host=spin.c07.dev.opsware.com
word.host=theword.c07.dev.opsware.com

ssl.keyPairs=<absolute path to spog.pkcs8>
ssl.trustCerts=<absolute path to opsware-ca.crt>
```

Distribution Directory

The following list shows what an expanded cbt-<version>.zip file contains.

```
% ls -R cbt
cbt:
bin      cfg      doc      filters  lib

cbt/bin:
cbt      cbtperm  rdql

cbt/cfg:
core.owl          java.policy      logging.template
version.txt
default.properties license.bea      opsware.owl
filter.owl        logging.bootstrap opsware.owl.keep

cbt/doc:
faq.html          install_config.html
index.html        rdf_flyer.64.gif

cbt/filters:
all.rdf           custext.rdf      os.rdf
servicelevel.rdf
app.rdf           distscript.rdf   patch.rdf
template.rdf     appconfigfile.rdf appconfig.rdf
compliancecriteria.rdf

cbt/lib:
DataAccess-14b.30.5.3.jar      icu4j.jar
DataAccess-14b.30.5.3.jar.inactive jakarta-oro-2.0.5.jar
HTTPClient14-hacked.jar      jena_0604.jar
antlr.jar                    junit.jar
bea-license.jar              opsware_common-1.0.5.jar
cbt.jar                      rdf-api-2001-01-19.jar
certicom-jdk14-wl700-patch.jar spinclient-14b.0.0.108.jar
common-1.2.0.jar             twistclient.jar
commons-logging.jar         weblogic.jar
concurrent.jar              xercesImpl.jar
copyright.txt               xml-apis.jar
ejb-2.0.jar
```


Chapter 2: Using the DCML Exchange Tool

IN THIS CHAPTER

This chapter includes the following topics:

- Introduction to the DCML Exchange Tool (DET)
- Exporting Content
- Export Filters
- Importing Content
- Policy on Importing Content Types
- Additional Considerations When Importing Customers
- Content Directory

Introduction to the DCML Exchange Tool (DET)

The DCML Exchange Utility (DET) is a tool that imports and exports Opsware content. The primary function of this tool is to provide a way to inject a newly-installed Opsware core with content. This tool can also be used to export partial content from one core and import it into other core instances.

Content, in the DET context, means user-created server management information in Opsware. This includes Customers, Applications, Patches, OS's, Service Levels, Templates, Custom Extensions, Distributed Scripts and associated information including Customer, MRLs, Install Hooks, Configuration Tracking Policies, Custom Attributes and Packages.

New content for DET 2.0 includes Application Configurations, Application Configuration Templates, and Server Compliance Criteria.

Content does not include managed environment type information. For example, facility information and server properties are not included. Also, CD&R, user information, user groups, and server groups are not included in this release of DET.

DET is a command-line utility that can be run on any Unix host with network connectivity to a target Opsware core. DET is written in Java and uses OWL and RDF for its schema definition and persistent store. DET imports and exports Opsware content by using Opsware component API's to extract both configuration and large binary content, such as packages and scripts.

DET Relationship to DCML

The content exported by the DCML Exchange Utility is in compliance with DCML Framework Specification v0.11, the first publicly-available specification of DCML. The DCML Exchange Tool uses a proprietary extension schema to describe contents exported from Opsware. The exported data.rdf is a valid DCML instance document that is parsable by a compliant DCML processor.

Exporting Content

DET exports the content you specify from a target Opsware core to an RDF/XML file that can be imported by DET into another Opsware core.

The export command is:

```
cbt -e <content_dir> -f <filter_file> -cf <target_core_config>
```

The command and its arguments indicate:

- `content_dir` - the path to a directory where the exported content will be stored. This directory will be created by the export function if it does not already exist.
- `filter_file` - a set of rules that tells DET what content it should export from the target Opsware core. See the "Export Filters" on page 11 for information on creating this file.
- `target_core_config` - a configuration file that tells DET where the various Opsware components are located, and what identity it should use to access them. Instructions for creating this file are found at "Installing and Configuring DCML Exchange Tool" on page 1.

The export command can be run multiple times using the same arguments, with the following caveats:

- If a filter has been specified, DET will ignore any previous exports in the content directory and will restart the export process.

- If a filter has not been specified, and the content directory contains a previously-aborted export, DET will pick up the export at the point where it was last aborted.

This checkpoint-restart feature is only available for the export command.

- If the export command specifies a content directory that contains a valid export (one which previously succeeded), DET will prompt the user if it OK to overwrite. If the user says it is not OK to overwrite, then DET will exit.



Before beginning an export or import process in a standalone core, shut down the Opsware Command Center to prevent users from changing any Opsware content until the process has completed.

In a multimaster mesh, first use the multimaster tools to make sure that the mesh is caught up and there are no conflicts, then shut down all Opsware Command Centers to prevent users from changing any Opsware content until the process has completed.

See the Opsware System Administration Guide for information about stopping and restarting the Opsware Command Center.

Export Filters

An export filter is a user-specified rule that tells DET what content to export – content that will subsequently be imported. Export filters are used with the following content types:

- Application Export Filter
- OS Export Filter
- Service Level Export Filter
- Template Export Filter
- Custom Extension Export Filter
- Distributed Script Export Filter
- Application Configuration Export Filter
- Application Configuration Template Export Filter
- Server Compliance Criteria Export Filter

Example Export Filter File

DET reads export filters in a specified filter file. The filter file is encoded in RDF/XML. The following is an example of a simple filter file that contains a single export filter rule.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE rdf:RDF [
3. <!ENTITY filter "http://www.opsware.com/ns/cbt/0.1/filter#">
4. ]>
5. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6.         xmlns="http://www.opsware.com/ns/cbt/0.1/filter#">
7. <ApplicationFilter rdf:ID="exportAppServers">
8.   <path>/Application Servers/Package Test</path>
9.   <directive rdf:resource="&filter;Descendants" />
10.</ApplicationFilter>
11.</rdf:RDF>
```

This example shows the standard filter headers in lines 1 through 6. These lines are the same in every filter, as is Line 11, which is the standard filter footer.

Lines 7 through 10 are the lines that are unique in each filter and indicate the specific function of the filter.

In the example above, there is just one export filter rule. However, filters can contain any number of unique filters between the standard header and footer lines. For example, this filter contains three export filter rules:

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE rdf:RDF [
3. <!ENTITY filter "http://www.opsware.com/ns/cbt/0.1/filter#">
4. ]>
5. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6.         xmlns="http://www.opsware.com/ns/cbt/0.1/filter#">
7. <ApplicationFilter rdf:ID="exportAppServers">
8.   <path>/Application Servers/Package Test</path>
9.   <directive rdf:resource="&filter;Descendants" />
10.</ApplicationFilter>
11.<ApplicationFilter rdf:ID="exportAppServfoo">
12.  <path>/Application Servers/Foo</path>
13.  <directive rdf:resource="&filter;Node"/>
14.</ApplicationFilter>
15.<CustomExtensionFilter rdf:ID="exportCustExtBulkPasswd">
16.  <scriptName>Bulk_Password_Changes</scriptName>
17.</CustomExtensionFilter>
18.</rdf:RDF>
```

Example filters can be found in the DET install directory under `<install_dir>/filters`.

This directory includes examples for each filter type and also an `all.rdf` filter, that exports all known Opsware data types from an Opsware core.

The following sections describe each filter type and their allowed parameters. In general, filter types map to an object type that can be manipulated by a user of the Opware Command Center. The Patch Filter, for example, maps to the Opware Command Center Patch object. Naming of the filters and their attributes also maps to the naming structure of the Opware Command Center so filter authors can quickly acquaint themselves with filters and their relevance to Opware content.

Application Export Filter

An application export filter tells DET what application nodes and associated content to export. The following application nodes are shown in the Opware Command Center by clicking the Software link in the navigation panel followed by the Application link on the Software menu.

- Application Servers
- Database Servers
- OS Extras
- Other Applications
- System Utilities
- Web Servers

The following tables describe the syntax of the Application Filter element:

Table 2-1: Application Export Filter Parameters

PARAMETER	DESCRIPTION
<code>rdf:ID</code>	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code> .

Table 2-2: Application Export Filter Nested Elements

ELEMENT	DESCRIPTION
<code>path</code> (required)	An absolute path from the top level of the software tree to the node to be exported. The path separator is <code>"/</code> .

Table 2-2: Application Export Filter Nested Elements

ELEMENT	DESCRIPTION
directive (required)	<p>An empty content element with a single <code>rdf:resource</code> parameter. The parameter refers to one of three constants:</p> <ul style="list-style-type: none"> • <code>Descendants</code> - export all descendants of the given path including the leaf of the path. • <code>Node</code> - only export the given node. • <code>Path</code> - export all nodes along the path and no other nodes. <p>For example, given the following path:</p> <pre>/Custom Applications/A/B/C/D</pre> <p>and your path is</p> <pre>/Custom Applications/A/B</pre> <p>If the <code>rdf:resource</code> parameter is <code>Node</code>, node B is exported.</p> <p>If the <code>rdf:resource</code> parameter is <code>Path</code>, nodes A and B are exported.</p> <p>If the <code>rdf:resource</code> parameter is <code>Descendants</code>, nodes B, C, and D are exported.</p>

Application Export Filter Examples

Export the /Application Servers/Foo node only.

```
<ApplicationFilter rdf:ID="exportAppServfoo">
  <path>/Application Servers/Foo</path>
  <directive rdf:resource="&filter;Node"/>
</ApplicationFilter>
```

Export Bar and Baz nodes along the given path. (Note that the stack root is not exported.)

```
<ApplicationFilter rdf:ID="exportDBServBarBaz">
  <path>/DBServer/Bar/Baz</path>
```

```

    <directive rdf:resource="&filter;Path"/>
</ApplicationFilter>

Export the patchtool node and all its descendants, including the leaf node.

<ApplicationFilter rdf:ID="exportSUpatchtool">
    <path>/System Utilities/patchtool</path>
    <directive rdf:resource="&filter;Descendants"/>
</ApplicationFilter>

```

Patch Export Filter

A patch export filter tells DET what patch or patch type to export.

Table 2-3: Patch Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code> .

Table 2-4: Patch Filter Nested Elements

ELEMENT	DESCRIPTION
<p>patchType (required)</p>	<p>A required nested element. This empty element has an <code>rdf:resource</code> parameter. This parameter may refer to one of the following patch types:</p> <ul style="list-style-type: none"> • APAR • APAR_FILESET • UPDATE_FILESET • AIX_Update_Fileset • HPUX_PATCH_PRODUCT • HPUX_Patch_Product • HPUX_PATCH_FILESET • HPUX_Patch_Fileset • SOL_PATCH • Solaris_Patch • SOL_PATCH_CLUSTER • Solaris_Patch_Cluster • HOTFIX • Windows_Hotfix • SERVICE_PACK • Windows_OS_Service_Pack • PATCH_META_DATA • Microsoft_Patch_Database
<p>patchName (optional)</p>	<p>An optional element that specifies the name of a specific patch. The name must be the patch unit_name, which is the name shown in the Opsware Command Center.</p>

Patch Filter Examples

Export the IY13260 APAR.

```
<PatchFilter rdf:ID="exportAPARIY13260">
  <patchName>IY13260</patchName>
  <patchType rdf:resource="&filter;APAR"/>
</PatchFilter>
```

Export all Solaris patches.

```
<PatchFilter rdf:ID="exportSolPatches">
  <patchType rdf:resource="&filter;SOL_PATCH"/>
</PatchFilter>
```

OS Export Filter

An Operating System export filter tells DET what Operating System node or Operating System type to export.

Table 2-5: OS Export Filter Parameters

PARAMETERS	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code> .

Table 2-6: OS Export Filter Nested Elements

ELEMENT	DESCRIPTION
osPlatform (required)	<p>A required nested element. This empty element has an <code>rdf:resource</code> parameter. This parameter may refer to one of the following operating systems:</p> <ul style="list-style-type: none"> • AIX_4.3 • AIX_5.1 • AIX_5.2 • AIX_5.3 • HP-UX_10.20 • HP-UX_11.00 • HP-UX_11.11 • Red_Hat_Enterprise_Linux_AS_2.1 • Red_Hat_Enterprise_Linux_AS_3 • Red_Hat_Enterprise_Linux_AS_3_X86_64 • Red_Hat_Enterprise_Linux_ES_2.1 • Red_Hat_Enterprise_Linux_ES_3 • Red_Hat_Enterprise_Linux_ES_3_X86_64 • Red_Hat_Enterprise_Linux_WS_2.1 • Red_Hat_Enterprise_Linux_WS_3_X86_64 • SuSE_Linux_8 • SuSE_Linux_Enterprise_Server_8 • SuSE_Linux_Enterprise_Server_9 • Red_Hat_Linux_6.2 • Red_Hat_Linux_7.1 • Red_Hat_Linux_7.2 • Red_Hat_Linux_7.3 • Red_Hat_Linux_8.0 • SunOS_5.6 • SunOS_5.7 • SunOS_5.8 • SunOS_5.9 • SunOS_5.10 • Windows_2000 • Windows_2003 • Windows_NT_4.0

OS Export Filter Examples

Export the "7.1 for mwp" Red Hat Linux 7.1 OS.

```
<OSFilter rdf:ID="exportOSRHLinux71">
  <osPlatform rdf:resource="&filter;Red_Hat_Linux_7.1"/>
  <osName>7.1 for mwp</osName>
</OSFilter>
```

Export all Solaris 5.6 OS's.

```
<OSFilter rdf:ID="exportOSSun56">
  <osPlatform rdf:resource="&filter;SunOS_5.6"/>
</OSFilter>
```

Service Level Export Filter

A service level export filter tells DET what service level nodes to export.

Table 2-7: Service Level Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code> .

Table 2-8: Service Level Export Filter Nested Elements

ELEMENT	DESCRIPTION
path (required)	An absolute path from the top level node to the node to be exported. The path separator is "/".

Table 2-8: Service Level Export Filter Nested Elements

ELEMENT	DESCRIPTION
<p><code>directive</code> (required)</p>	<p>An empty content element with a single <code>rdf:resource</code> parameter. The parameter refers to one of three constants:</p> <ul style="list-style-type: none"> • Descendants -export all descendants of the given path including the leaf of the path. • Node - only export the given node. • Path - export all nodes along the path and no other nodes. <p>For example, given the following path:</p> <pre>/Service Levels/A/B/C/D</pre> <p>and your path is</p> <pre>/Service Levels/A/B</pre> <p>If the <code>rdf:resource</code> parameter is Node, node B is exported.</p> <p>If the <code>rdf:resource</code> parameter is Path, nodes A and B are exported.</p> <p>If the <code>rdf:resource</code> parameter is Descendants, nodes B, C and D are exported.</p>

Service Level Export Examples

Export the /Service Levels/Foo node only.

```
<ServiceLevelsFilter rdf:ID="exportServLevfoo">
  <path>/Service Levels/Foo</path>
  <directive rdf:resource="&filter;Node"/>
</ServiceLevelFilter>
```

Export Bar and Baz nodes along the given path. Note that the stack root is not exported.

```
<ServiceLevelFilter rdf:ID="exportServLevBarBaz">
  <path>/ServiceLevels/Bar/Baz</path>
```

```

    <directive rdf:resource="&filter;Path"/>
</ServiceLevelFilter>

Export the Gold Service level node and all of its descendants, including the leaf node.

<ServiceLevelFilter rdf:ID="exportServLevGold">
    <path>/ServiceLevels/Gold</path>
    <directive rdf:resource="&filter;Descendants"/>
</ServiceLevelFilter>

```

Template Export Filter

A template export filter tells DET what template nodes to export.

Table 2-9: Template Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code> .

Table 2-10: Template Export Filter Nested Elements

ELEMENT	DESCRIPTION
path (required)	An absolute path from the top level node to the node to be exported. The path separator is "/".

Table 2-10: Template Export Filter Nested Elements

ELEMENT	DESCRIPTION
<p><code>directive</code> (required)</p>	<p>An empty content element with a single <code>rdf:resource</code> parameter. The parameter refers to one of three constants:</p> <ul style="list-style-type: none"> • Descendants - export all descendants of the given path including the leaf of the path. • Node - only export the given node. • Path - export all node along the path and no other nodes. <p>For example, given the following path:</p> <pre>/Templates/A/B/C/D</pre> <p>and your path is</p> <pre>/Templates/A/B</pre> <p>If the <code>rdf:resource</code> parameter is <code>Node</code>, node B is exported.</p> <p>If the <code>rdf:resource</code> parameter is <code>Path</code>, nodes A and B are exported.</p> <p>If the <code>rdf:resource</code> parameter is <code>Descendants</code>, nodes B, C, and D are exported.</p>

Template Export Filter Examples

Export the `/Templates/Foo` node only.

```
<TemplateFilter rdf:ID="exportTemplatesfoo">
  <path>/Templates/Foo</path>
  <directive rdf:resource="&filter;Node"/>
</TemplateFilter>
```

Export Bar and Baz nodes along the given path. Note that the stack root is not exported.

```
<TemplateFilter rdf:ID="exportTemplatesBarBaz">
  <path>/Templates/Bar/Baz</path>
```

```

    <directive rdf:resource="&filter;Path"/>
</TemplateFilter>

```

Export the Alpha Template node and all of its descendants, including the leaf node.

```

<TemplateFilter rdf:ID="exportTemplatesAlpha">
    <path>/Templates/Alpha</path>
    <directive rdf:resource="&filter;Descendants"/>
</TemplateFilter>

```

Custom Extension Export Filter

A custom extension export filter tells DET to either export a specific custom extension or all custom extensions. If you want to export more than one custom extension, but not all, create a filter for each custom extension you want to export.

Table 2-11: Custom Extension Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code> .

Table 2-12: Custom Extension Export Filter Nested Elements

ELEMENT	DESCRIPTION
scriptName (optional)	An optional element that specifies a script to export. The script name does not include the account prefix. If this element is omitted, all custom extension scripts are exported.

Custom Extension Export Filter Examples

Export the Bulk_Password_Changes custom extension script only.

```

<CustomExtensionFilter rdf:ID="exportCustExtBulkPasswd">

```

```

    <scriptName>Bulk_Password_Changes</scriptName>
</CustomExtensionFilter>
Export all custom extension scripts.
<CustomExtensionFilter rdf:ID="exportAllCustExtScripts"/>

```

Distributed Script Export Filter

A distributed export script filter tells DET to either export a specific distributed script or all distributed scripts. Only shared distributed scripts are exported and imported.

Table 2-13: Distributed Script Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code>

Table 2-14: Distributed Script Export Filter Nested Elements

ELEMENT	DESCRIPTION
scriptName (optional)	An optional element that specifies a script to export. The script name does not include the <code>__global__</code> prefix. If this element is omitted, all shared distributed scripts are exported.

Distributed Script Export Filter Examples

Export the shared ls distributed script only.

```

<DistributedScriptFilter rdf:ID="exportScriptLS">
    <scriptName>ls</scriptName>
</DistributedScriptFilter>

```

Export all shared distributed scripts.


```
<DistributedScriptFilter rdf:ID="exportAllSharedScripts"/>
```

Application Configuration Export Filter

The Application Configuration export filter tells DET what Application Configurations you want to export. An Application Configuration is a container for one or more Application Configuration Template files. Thus, if you export an Application Configuration, you will also be exporting all template files inside it.

Table 2-15: Application Configuration Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code>

Table 2-16: Application Configuration Export Filter Nested Elements

ELEMENT	DESCRIPTION
configurationName (optional)	An optional element that specifies the name of the Application Configuration. Use this if you want to export specific Application Configurations by name.
customerName (optional)	An optional element that specifies to export all Application Configurations that have been associated with the specified customer.
osPlatform rdf:resource (optional)	An optional element that specifies to export all Application Configurations that have been associated with the specified OS.

Application Configuration Export Filter Example

Export all Application Configurations.

```
<ApplicationConfigurationFilter rdf:ID="getAllAppConfigs"/>
```

Export only the Application Configuration named “iPlanet” that is customer independent and that has been associated with the SunOS 5.8 operating system.

```
<ApplicationConfigurationFilter rdf:ID="getSpecificAppConfigs">
  <configurationName>iPlanet</configurationName>
  <customerName>Customer Independent</customerName>
  <osPlatform rdf:resource="&filter;SunOS_5.8"/>
</ApplicationConfigurationFilter>
```

Application Configuration Template Export Filter

The Application Configuration Template export filter tells DET what Application Configuration Template files you want to export.

Table 2-17: Application Configuration Template Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format <code>rdf:ID="unique name"</code>

Table 2-18: Application Configuration Template Export Filter Nested Elements

ELEMENT	DESCRIPTION
configurationFileName (optional)	An optional element that specifies the name of the Application Configuration Template. Use this if you want to export specific Application Configuration Templates by name.
osPlatform rdf:resource (optional)	An optional element that specifies to export all Application Configurations that have been associated with the specified OS.

Table 2-18: Application Configuration Template Export Filter Nested Elements

ELEMENT	DESCRIPTION
customerName (optional)	An optional element that specifies to export all Application Configuration Templates that have been associated with the specified customer.

Application Configuration Template Export Filter Examples

Export all Application Configuration Templates.

```
<ApplicationConfigurationFileFilter
rdf:ID="getAllAppConfigTemps"/>
```

Export the specific Application Configuration Template named "iplanet6.1_mimetypes.tpl" that is customer independent and is associated with the Red Hat Enterprise Linux AS 3 X86_64 operating system.

```
<ApplicationConfigurationFileFilter
rdf:ID="getSpecificAppConfigTemp">
  <configurationFileName>iplanet6.1_mimetypes.tpl</
configurationFileName>
  <customerName>Customer Independent</customerName>
  <osPlatform rdf:resource="&filter;Red_Hat_Enterprise_
Linux_AS_3_X86_64"/>
</ApplicationConfigurationFileFilter>
```

Server Compliance Criteria Export Filter

The Server Compliance Criteria export filter tells DET what Server Compliance Criteria you want to export.

Table 2-19: Server Compliance Criteria Export Filter Parameters

PARAMETER	DESCRIPTION
rdf:ID	Each filter has a unique name that is specified in the filter file using the format rdf:ID="unique name"

Table 2-20: Server Compliance Criteria Filter Nested Elements

<code>selectionCriteriaName (optional)</code>	<p>An optional element that specifies the name of the Server Compliance Criteria. Use this if you want to export specific Server Compliance Criteria by name.</p>
<code>osType rdf:resource (optional)</code>	<p>An optional element that specifies to export all Server Compliance Criteria that have been associated with the specified OS. The two possible values for this element are either Windows or Unix. For example,</p> <pre><osType rdf:resource="&filter;Windows" /></pre> <p>or</p> <pre><osType rdf:resource="&filter;Unix" /></pre>

Server Compliance Criteria Export Filter Examples

Export all Server Compliance Criteria.

```
<ComplianceSelectionCriteriaFilter
rdf:ID="getAllSelectionCriteria"/>
```

Export the specific Server Compliance Criteria named "My Selection Criteria" that has been associated with the Windows operating system.

```
<ComplianceSelectionCriteriaFilter
rdf:ID="getSpecificSelectionCriteria">
  <selectionCriteriaName>My Selection Criteria</
selectionCriteriaName>
  <osType rdf:resource="&filter;Windows" />
</ComplianceSelectionCriteriaFilter>
```

Importing Content

The Import process imports content to a target Opsware core.



Content import using the DET into an Opsware core is supported on a forward compatible basis. For example, you can import content from an Opsware System 4.7 core into an Opsware System 5.0 core. (But you cannot do this in reverse.)

The import command is:

```
cbt -i <content_dir> -p <policy> -cf <target_core_config>
```

The command and its arguments indicate:

- `content_dir` - the directory containing the previously-exported content
- `policy` - the import policy that DET should use when it detects duplicates in the target Opsware core. See the “Policy on Importing Content Types” on page 29.
- `target_core_config` - a configuration file that tells DET where the various Opsware components are located, and what identity it should use to access them. Instructions for creating this file are located at “Installing and Configuring DCML Exchange Tool” on page 1.

See “DCML Exchange Tool Command Line” on page 37 for a complete list all the available arguments and their meanings.

When Applications are imported using DET, the associated package name in the Opsware Core receives a “cbt” suffix. For example:

```
openssh-3.8p1-sol8-sparc-local_cbt796213986
```

Policy on Importing Content Types

The following table shows the affect of the policy you specify on the command line for each content type when duplicates are found.

The choices are:

- `overwrite` - the default if no policy is specified. The effect of this option is different for each content type as described in the table.
- `duplicate` - the effect of this option is different for each content type as described in the following table.

- skip - for all content types, specifying “skip” means that if a duplicate is found, a message is entered in the session log and the import continues.

See “DCML Exchange Tool Command Line” on page 37 for a complete list of all the available arguments and their meanings.

Table 2-21: Policies Used By DET When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
Application	<ul style="list-style-type: none"> • Custom attributes • Config tracking policy • Install order • Software list • Customer 	<p>Content information overrides existing node in target Opsware core without changing its node ID.</p> <p>Content information is overlaid on the existing node.</p>	<p>Content information is renamed by applying a “cbt- <random>” suffix to the application name.</p>
Patch	NA	<p>Physical patch package is uploaded and contained units are created in the Software Repository. In case of AIX LPPs and HPUX Depots, the package is renamed by suffixing a “cbt- <random>” string. NOTE: this causes duplicate unit information to be displayed in the OCC.</p>	<p>Same as overwrite. This is because the Opsware system cannot reliably and efficiently determine whether a package in the Software Repository is equivalent to the package being uploaded.</p>

Table 2-21: Policies Used By DET When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
Patch Knowledge (PATCH_META_DATA)	NA	The patch database is imported into the Opsware system, overwriting the existing database, if there is one. The knowledge created by the import will depend on the patch preference settings in the target Opsware core.	Same as overwrite.
OS	<ul style="list-style-type: none"> • Custom attributes • Config tracking policy • Customer • Software list • InstallHooks • MRL 	Content information overrides existing node in target Opsware core without changing its node ID. Content information is overlaid on the existing node.	Content information is renamed by applying a “cbt- <random>” suffix to the application name.
Template	<ul style="list-style-type: none"> • Custom attributes • Customer • Application • Patch • OS • Service Level 	Content information overrides existing node in target Opsware core without changing its node ID. Content information is overlaid on the existing node.	Content information is renamed by applying a “cbt- <random>” suffix to the template name.

Table 2-21: Policies Used By DET When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
Service Level	<ul style="list-style-type: none"> • Custom attributes • Config tracking policy • Customer 	Content information overrides existing node in target Opsware core without changing its node ID. Content information is overlaid on the existing node.	Content information is renamed by applying a “cbt-<random>” suffix to the template name.
Distributed Script	NA	A new version of the script is created.	Same as overwrite policy.
Custom Extension	NA	A new version of the script is created.	Same as overwrite policy.
Customer	NA	Do nothing on duplication.	Do nothing on duplication. Please see “Additional Considerations When Importing Customers” on page 34 for important information about importing customers.
Package	NA	Package is uploaded over the existing package. For LPP and DEPOT package types, the package is suffixed with “cbt<random integer>”. For Solaris package types, the package is suffixed with “cbt<random int>” unless it's of file type “.tar”. In which case, the package file is suffixed with “cbt<random int>.tar”.	Same as overwrite. This is because it cannot reliably and efficiently determined whether a package in the Software Repository is equivalent to the package being uploaded.

Table 2-21: Policies Used By DET When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
MRL	NA	Always create an MRL with the name "Synthetic OS Media" or associate it with the existing instance of this MRL.	Same as overwrite.
Install Hooks	NA	See Unit.	See Unit.
Unit	Unit script	Units are associated with a physical package, see Package content type above. Virtual units are always associated with existing units in the target Opsware core - this is presumably created as a side effect of uploading the physical package that is also part of the same import session.	Same as overwrite.
Install Order Relationship	NA	Creates the relationship regardless and override the existing relationship.	Since this is done in the context of the parent node, a new relationship is always created because a parent node is always created - albeit with a different name.
Custom Attributes	NA	Creates and overrides existing keys. The result is the union of the imported key and existing keys.	Same as Install Order Relationship.
Software List	Unit	Creates and overrides existing list.	Same as Install Order Relationship.

Table 2-21: Policies Used By DET When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
Config Tracking Policy	NA	Creates and overrides existing policy.	Same as Install Order Relationship.
Unit Script	NA	Created and overrides existing unit scripts.	Same as overwrite.
Application Configuration	Account Application Configuration File	All attributes are updated in overwrite mode.	New Application Configuration is created and named "Oldname-cbt- <random>"
Application Configuration Template	Account	All attributes are updated in overwrite mode.	New Application Configuration template is created and named "Oldname-cbt- <random>"
Server Compliance Criteria	NA	All attributes are updated in overwrite mode.	New Server Compliance Criteria is created and named "Oldname-cbt- <random>"

Additional Considerations When Importing Customers

Currently, DET does not support the export of user group permissions that are associated with customers, except in cases when the customer name being exported has the same name as a customer in the target core (the core you are importing the customer into).

For example, let's say that in your source core, you had a software application node named iPlanet, and that software application node iPlanet was accessible for reading and writing to all groups associated with a customer named Computing Machines. One of these groups associated with the customer Computing Machines was named groupA.

Next, you export a software application node iPlanet from the source core, and then import that node into a new core – and this core does not have a customer named Computing Machines. The result would be that any users in groupA would not be able to see software application node iPlanet in the target core.

However, if the core you imported the customer Computing Machines into already has a customer with exactly the same name, then all permissions are untouched in the new core and all users groupA would be able to access the software application node named iPlanet – in other words, all permissions associated with the Computing Machines customer (the ability to read and write the software node iPlanet) will remain in tact.

Importing Customers Workaround

If a user group loses permissions to access objects (such as servers associated with a customer), then use the Opsware Command Center to re-assign the permissions. Until doing so, only users who are administrators will see these customers and their associated objects.

Content Directory

The content directory is the persistent store of exported Opsware content. The content directory contains:

- `data.rdf` - a database of exported Opsware configuration content.
- `filter.rdf` - a database of filters provided by the user and generated by DET.
- `blob/` - a directory containing exported software packages and scripts.
- `var/` - a directory containing logs for each of the last ten import and export sessions. Logs are named `cbtexport {0-9}.log` and `cbtimport {0-9}.log`. The 0 log is always the most recent and the 9 log file is always the oldest of the ten session logs.

The following is an example content directory.

```
% ls -R
.:
blob
data.rdf
filter.rdf
var
```

```
./blob:  
unitid_140270007.pkg  
unitid_166510007.pkg  
unitid_166540007.pkg  
unitid_2090007.pkg
```

```
./var:  
cbtexport0.log  
cbtexport0.log.lck  
cbtimport0.log
```

Appendix A: DCML Exchange Tool Command Line

IN THIS APPENDIX

This Appendix contains the following topics:

- About DCML Exchange Tool (DET) Command Line
- Export Command
- Import Command
- Show Export Status Command
- Configuration File Command
- Show Version Command
- Show Help Command
- DET Permissions Command

About DCML Exchange Tool (DET) Command Line

The DET command line is pre-configured to be executed as the user root on a managed server. If used in this configuration, you will only have to provide your Opsware user name and password to perform an export or an import. The following is an example session: (The example below assumes the user has been granted import and export permission. See “Installing and Configuring DCML Exchange Tool” on page 1 in Chapter 1 for more information.)

The following is an example csh session on the Opsware Command Center server.

```
% setenv JAVA_HOME <j2re 1.4.x installation>
% <cbt install dir>/bin/cbt -e /tmp/foo -f <cbt install dir>/filters/app.rdf --spike.username
hermaime
Enter password for hermaime: *****
...
```

Commands and Options

The following sections describe the syntax of the DET command line interface.

Export Command

The export command uses the following syntax:

```
cbt -e <content_dir> [-f <filter_file>] [-cf file] [-d] [-np]
```

Table A-1: Export Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-e	--export	Export Opsware data from an Opsware core and store the data in the given content directory.
-f	--filter	The first time you export, you must specify a filter describing what data to export. After that, if no filter is specified, then any previously-used filter in the content directory is used.
-c	--clean	Remove previously exported data from the given content directory.
-cf	--config	Read configuration from the given file.
-d	--debug	Show more detailed debug information.
-np	--noprogess	Don't show the progress on the console.

Import Command

The import command uses the following syntax:

```
cbt -i <content_dir> [-p <overwrite|duplicate|skip>] [-cf file] [-d] [-np]
```

Table A-2: Import Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-i	--import	Import Opsware data from the given content directory.
-p	--policy	Import policy. Default is "overwrite." "overwrite" means to override objects in the same name space on the target Opsware system without affecting its object IDs. "duplicate" means to create a duplicate copy of an object with a synthetic name when a duplicate is detected on the target Opsware system. "skip" is the most conservative policy. It aborts the import of an object if the same object is detected in the target Opsware system.
-cf	--config	Read configuration from the given file.
-d	--debug	Show more detailed debug information.
-np	--noprogress	Don't show the progress on the console.

Show Export Status Command

The show export status command uses the following syntax:

```
cbt -t <content_dir>
```

Table A-3: Show Export Status Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-t	--showstatus	Show status of export of the given content directory.

Configuration File Command

The configuration file command option uses the following syntax:

```
cbt -s [-cf config]
```

Table A-4: Configuration File Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-s	--showconfig	Show current configuration values.
-cf	--config	Read configuration from the given file.

Show Version Command

The show version command uses the following syntax:

```
cbt -v
```

Table A-5: Show Version Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-v	--version	Show the version of the DET tool.

Show Help Command

The show help command uses the following options:

```
cbt -h
```

Table A-6: Show Help Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-h	--help	Display this help message.

DET Permissions Command

The DET permissions command uses the following syntax:

```
cbtperm -u [user] -a [spike.username] -p [spike.port] -s [spike.host] -c [ssl.trustCerts] -k [ssl.keyPairs]
```

Table A-7: DET Permissions Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-u	N/A	The user to whom you want to grant permission to use the DCML Exchange Tool.
-a	--spike.username	User name for Spike authentication, such as the Opsware administrator.
-p	--spike.port	Spike's listener port.
-s	--spike.host	Spike's hostname or IP.

Table A-7: DET Permissions Command Options

SHORT OPTION	LONG OPTION	DESCRIPTION
-c	--ssl.trustCerts	Comma-separated list of trust certificate files to be used to communicate with XML-RPC servers
-k	--ssl.keyPairs	Comma-separated list of client certificates to be used to communicate with XML-RPC servers