Opsware® System DCML Exchange Utility 1.2 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

Table of Contents

Preface

Welcome to the Opsware DCML Exchange Utility (DEU.) 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.

The content that can be exported includes:

- Applications
- Patches
- Operating Systems
- Templates
- Service Levels
- Distributed Scripts
- Custom Extensions

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 Utility, starting with an introduction. It 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 Utility - provides information about the installation and configuration of the DCML Exchange Utility, 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 Utility - 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: The 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
Bold	Defines terms.
Italics	Identifies guide titles and provides emphasis.
`Courier`	Identifies text of displayed messages and other output from Opsware programs or tools.
`Courier Bold`	Identifies user-entered text (commands or information).
`Courier Italics`	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 Utility

IN THIS CHAPTER

This chapter describes:

- Installing and Configuring DEU

Installing and Configuring DEU

The DCML Exchange Utility (DEU) can be run on any UNIX machine, not necessarily a managed server. The following instructions detail how to install and configure the DCML Exchange Utility.

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 for that information.
4. Set your JAVA_HOME environment variable to point to a Java 1.4.x installation by issuing the following command:
   ```
   % setenv JAVA_HOME <j2re 1.4.x installation>
   ```
5. Perform the following steps for each core that DEU 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:
      ```
      % setenv JAVA_HOME <j2re 1.4.x installation>
      % 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
Utility.
%
```

3. Obtain a copy of the opsware-ca.crt trust certificate from
   `/var/lc/crypto/twist/opsware-ca.crt`
   and save it in a location DEU can access. This step is optional if you are running DEU from the server where the Opsware Command Center is installed.
4. Obtain a copy of the spog.pkcs8 client certificate from
   `/var/lc/crypto/twist/spog.pkcs8`
   and save it in a location DEU can access. This step is optional if you are running DEU 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 DEU. This configuration file contains Opsware component access information that would otherwise need to be given on the DEU command-line. The core configuration file is a key=value pair text file.

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

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
`cbt.numthreads`	`1`	Number of concurrent threads used for import and export.
`twist.port`	`1026`	Web Services Data Access Engine's listening port.
`twist.host`	`localhost`	Web Services Data Access Engine's host name or IP.
`twist.protocol`	`iiop`	Web Services Data Access Engine's protocol. This should be t3 or t3s.
`twist.username`	`buildmgr`	Web Services Data Access Engine's username (for example, “buildmgr”).
`twist.password`	`buildmgr`	Web Services Data Access Engine's password.
`twist.certpaths`	`/var/lc/crypto/twist/opsware-ca.crt`	Comma-separated list of trust certificates used to communicate with the Web Services Data Access Engine.
`way.port`	`1018`	Command Engine's listener port.
`way.host`	`way`	Command Engine's host name or IP.
`way.protocol`	`https`	Command Engine's listener protocol. This is typically HTTPS.
`way.path`	`wayrpc.py`	Command Engine's base URL path.

Table 1-1: Configuration Properties

PROPERTY NAME	DEFAULT VALUE	DESCRIPTION
`spin.port`	`1004`	Data Access Engine's listener port.
`spin.host`	`spin`	Data Access Engine's host name or IP.
`spin.protocol`	`http`	Data Access Engine's listener protocol. HTTP if the DEU 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.
`spin.path`	`spinrpc.py`	Data Access Engine's base URL path.
`word.port`	`1003`	Software Repository's listener port.
`word.host`	`word`	Software Repository's host name or IP.
`word.protocol`	`https`	Software Repository's listener protocol. This is HTTPS.
`word.path`	`wordrpc.py`	Software Repository's base URL path.
`spike.port`	`1018`	Spike's listener port.
`spike.host`	`way`	Spike's host name or IP.
`spike.protocol`	`https`	Spike's listener protocol. This is typically HTTPS.
`spike.path`	`wayrpc.py`	Spike's base URL path.
`spike.enabled`	`true`	Use Spike for authentication and authorization on all XML-RPC-based servers.

Table 1-1: Configuration Properties

PROPERTY NAME	DEFAULT VALUE	DESCRIPTION
`spike.username`	`admin`	User name for Spike authentication. This is the user who was granted permissions by the cbtperm tool. Default is the Opsware administrator.
`spike.password`	`<no default>`	User password for Spike authentication. This is an OCC user's password.
`ssl.useHttpClient`	`true`	Use the HTTPClient library instead of JDK's built-in HTTP client.
`ssl.trustCerts`	`/var/lc/crypto/twist/opsware-ca.crt`	Comma-separated list of trust certificate files used to communicate with XML-RPC-based servers.
`ssl.keyPairs`	`/var/lc/crypto/twist/spog.pkcs8`	Comma-separated list of client certificates used to communicate with XML-RPC-based servers.

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=1031
twist.protocol=t3s
twist.username=<twist_user>
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

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 Utility

IN THIS CHAPTER

This chapter includes:

Introduction to the DCML Exchange Utility

The DCML Exchange Utility (DEU) 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 DEU 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. Content does not include managed environment type information. For example, facility information and server properties are not included. Also, CD&R and user information are not included in this release of DEU.

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

DEU 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. The DCML Exchange Utility 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

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

The export command is:

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

If the PATH has not been set, the command is ./cbt, followed by the desired arguments. See “Installing and Configuring DCML Exchange Utility” on page 1 in Chapter 1 for more information about setting the PATH.

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 DEU what content it should export from the target Opsware core. See the “Export Filters” on page 9 for information on creating this file.
- `target_core_config` - a configuration file that tells DEU 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 Utility” 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, DEU 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, DEU will pick up the export at the point where it was last aborted.

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

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 DEU what content to export– content that will subsequently be imported. Export filters are used with the following content types:

- Application
- Patch
- OS
- Template
- Service Level
- Distributed Script
- Custom Extension

Example Export Filter File

DEU 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.          xlmns=”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 exporet 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.          xlmns=”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 DEU 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 Opsware Command Center. The Patch Filter, for example, maps to the Opsware Command Center

Patch object. Naming of the filters and their attributes also maps to the naming structure of the Opsware Command Center so filter authors can quickly acquaint themselves with filters and their relevance to Opsware content.

Application Filter

An application filter tells DEU what application nodes and associated content to export. The following application nodes are shown in the Opsware 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 Utilties
- Web Servers

Table 2-2: Application 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 2-3: Application Filter Nested Elements

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

Table 2-3: Application Filter Nested Elements

ELEMENT	DESCRIPTION
`directive (required)`	An empty content element with a single `rdf:resource` parameter. The parameter refers to one of three constants: • `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. For example, given the following path: `/Custom Applications/A/B/C/D` and your path is `/Custom Applications/A/B` If the `rdf:resource` parameter is Node, node B is exported. If the `rdf:resource` parameter is Path, nodes A and B are exported. If the `rdf:resource` parameter is Descendants, nodes B, C, and D are exported.

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 Filter

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

Table 2-4: Patch 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 2-5: Patch Filter Nested Elements

ELEMENT	DESCRIPTION
`patchType (required)`	A required nested element. This empty element has an `rdf:resource` parameter. This parameter may refer to one of the following patch types: • 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
`patchName (optional)`	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.

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 Filter

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

Table 2-6: OS Filter Parameters

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

Table 2-7: OS Filter Nested Elements

ELEMENT	DESCRIPTION
`osPlatform (required)`	A required nested element. This empty element has an `rdf:resource` parameter. This parameter may refer to one of the following Operating Systems: • AIX_4.3 • AIX_5.1 • AIX_5.2 • 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_ES_2.1 • Red_Hat_Enterprise_Linux_ES_3 • SuSE_Linux_Enterprise_Server_8 • 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 • Windows_2000 • Windows_2003 • Windows_NT_4.0

Table 2-7: OS Filter Nested Elements

ELEMENT	DESCRIPTION
`osName Filter (optional)`	An optional elment that specifies the name of a specific Operating System. The name must be the same as the name displayed in the Opsware Command Center.

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 OSs.

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

Service Level Filter

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

Table 2-8: Service Level 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 2-9: Service Level 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 2-9: Service Level Filter Nested Elements

ELEMENT	DESCRIPTION

`directive (required)`	An empty content element with a single `rdf:resource` parameter. The parameter refers to one of three constants:
	• 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.
	For example, given the following path:
	`/Service Levels/A/B/C/D`
	and your path is
	`/Service Levels/A/B`
	If the `rdf:resource` parameter is Node, node B is exported.
	If the `rdf:resource` parameter is Path, nodes A and B are exported.
	If the `rdf:resource` parameter is Descendants, nodes B, C and D are exported.

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 Filter

A template filter tells DEU what template nodes to export.

Table 2-10: Template 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 2-11: Template 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 2-11: Template Filter Nested Elements

ELEMENT	DESCRIPTION

`directive (required)`	An empty content element with a single `rdf:resource` parameter. The parameter refers to one of three constants:
	• 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.
	For example, given the following path:
	`/Templates/A/B/C/D`
	and your path is
	`/Templates/A/B`
	If the `rdf:resource` parameter is Node, node B is exported.
	If the `rdf:resource` parameter is Path, nodes A and B are exported.
	If the `rdf:resource` parameter is Descendants, nodes B, C, and D are exported.

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 Filter

A custom extension filter tells DEU 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 2-12: Custom Extension 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 2-13: Custom Extension 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.

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 Filter

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

Table 2-14: Distributed Script 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 2-15: Distributed Script Filter Nested Elements

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

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"/>
```

Importing Content

The Import process imports content to a target Opsware core.

The import command is:

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

If the PATH has not been set, the command is ./cbt, followed by the desired arguments. See "Installing and Configuring DCML Exchange Utility" on page 1 in Chapter 1 for more information about setting the PATH.

The command and its arguments indicate:

- `content_dir` - the directory containing the previously-exported content
- `policy` - the import policy that DEU should use when it detects duplicates in the target Opsware core. See the "Policy on Importing Content Types" on page 23.
- `target_core_config` - a configuration file that tells DEU 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 Utility" on page 1.

See "DCML Exchange Utility Command Line" on page 29 for a complete list of all the available arguments and their meanings.

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 typs, specifying "skip" means that if a duplicate is found, a message is entered in the session log and the import continues.

SSee “DCML Exchange Utility Command Line” on page 29 for a complete list of all the available arguments and their meanings.

Table 2-16: Policies Used By DEU When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
Application	• Custom attributes • Config tracking policy • Install order • Software list • 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 application name.
Patch	NA	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.	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.
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.

Table 2-16: Policies Used By DEU When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
OS	• 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	• 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.
Service Level	• 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.

Table 2-16: Policies Used By DEU When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
Customer	NA	Do nothing on duplication.	Do nothing on duplication.
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.
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.

Table 2-16: Policies Used By DEU When Importing Each Content Type

CONTENT TYPE	ASSOCIATED CONTENT TYPES	IMPORT POLICY (OVERWRITE)	IMPORT POLICY (DUPLICATE)
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.
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.

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 DEU.
- `blob/` - a directory containing exported software packages and scripts.
- `var/` - a directory containing logs for each of the last five sessions. Logs are named cbt{0-4}.log. The cbt0.log is always the most recent and the cbt4.log file is always the oldest of the five 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:
cbt0.log
cbt0.log.lck
cbt1.log
cbt2.log
cbt3.log
cbt4.log
```

Appendix A: DCML Exchange Utility Command Line

IN THIS APPENDIX

This Appendix describes how to use the DEU Command line and contains the following topics:

- DEU Command Line

DEU Command Line

The DEU command line is pre-configured to be executed as the user root on an Opsware-managed server. If used in this configuration, the user will only have to provide his Opsware Command Center username and password in order 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 Utility" on page 1 in Chapter 1 for more information.)

The following is an example shell 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: ********
...
```

The following table shows the options available for the DEU command-line interface and a description of each.

Table I-17: DEU Command Line Options and Descriptions

DEU COMMAND LINE ENTRY		**DESCRIPTION**
SHORT	**LONG**	
Export Command `Usage: cbt -e <content_dir> [-f <filter_file>] [-cf file] [-d] [-np] ...`		
`-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.
`-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.
Import Command `Usage: cbt -i <content_dir> [-p <overwrite|duplicate|skip>] [-cf file] [-d] [-np] ...`		
`-i`	`--import`	Import Opsware data from the given content directory.

Table I-17: DEU Command Line Options and Descriptions

DEU COMMAND LINE ENTRY		DESCRIPTION
SHORT	LONG	
`-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.
Clean Content Directory Command `Usage: cbt -c <content_dir>`		
`-c`	`--clean`	Remove previously exported data from the given content directory.
Show Export Status Command `Usage: cbt -t <content_dir>`		
`-t`	`--showstatus`	Show status of export of the given content directory.
Configuration File Command `Usage: cbt -s [-cf config] ...`		
`-s`	`--showconfig`	Show current configuration values.
`-cf`	`--config`	Read configuration from the given file.
Show Version Command `Usage: cbt -v`		
`-v`	`--version`	Show the version of the DEU tool.

Table I-17: DEU Command Line Options and Descriptions

DEU COMMAND LINE ENTRY		DESCRIPTION
SHORT	LONG	
Show Help Command `Usage: cbt -h`		
`-h`	`--help`	Display this help message.
DEU Permissions Command `Usage: cbtperm -u [user] -a [spike.username] -p [spike.port] -s [spike.host] -c [ssl.trustCerts] -k [ssl.keyPairs]`		
`-u`	`N/A`	The user to whom you want to grant permission to use the DCML Exchange Utility.
`-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 host name or IP.
`-s`	`--ssl.trustCerts`	Comma-separated list of trust certificate files used to communicate with XML-RPC-based servers.
`-k`	`--ssl.keyPairs`	Comma-separated list of client certificates used to communicate with XML-RPC-based servers.