HP OpenView Configuration Management

Batch Publisher

for UNIX and Windows operating systems

Installation and Configuration Guide

Software Release Date: April 2007

Document Release Date: April 2007

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Trademark Notices

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates or to verify that you are using the most recent edition, visit the following URL:

ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Table 1 lists new features added for the Configuration Management v 5.0 release.

Table 1 New features added for Configuration Management 5.0

Chapter	Version	Changes
1	5.00	Page 17, System Requirements and Availability: updated for version 5.00.
5	5.00	Page 68, CM Native Packaging Command-Line Interface: Added –depth and -dist options.
5	5.00	Page 80, Publishing with Interactive Mode: Updated to include new continue option behavior.
5	5.00	Page 94, Operational Notes: Included notes for new command line options and multi-level dependency support.
5	5.00	Page 93, Automatic Inclusion of Required Packages: Updated information for default behavior and RPM packages.
5	5.00	NOCHECK information added to Operational Notes and class instance tables.

Table 2 indicates changes made to this document for earlier releases.

Table 2 Document changes

Chapter	Version	Changes
5	3.1	Page 69, Table 9: added new command-line parameters: -coreq, -I, -M, -S
5	3.1	Page 80, Publishing with Interactive Mode: new section.
5	3.1	Page 82, Table 12: added CONTENTS attribute for SD class.
5	3.1	Added new SVR4 class instance attributes: ADMIN, AMDINOBJ, CONTENTS, PKGVER, PKGREV.
5	3.1	Added new RPM class instance attributes: PKGVER, PKGREL, CONTENTS, PKGEPOCH.
5	3.1	Page 93, Automatic Inclusion of Required Packages: new section.
5	3.1	Page 94, Operational Notes: new section.
1	3.0	Added a note about the use of prefixes in service names.
3	3.0	Added **replacepkg**
4	3.0	Removed ZPROMOTE section.
5	3.01	Information about AIX, SOLPATCH, RedHat Linux, and RPM was added throughout this chapter.
5	3.0	Added a note about publishing native packages to the end of the *Overview* section.
5	3.01	Added Required Classes section.
5	3.0	Added table Command Line Usage for RNP.

Support

Please visit the HP OpenView support web site at:

www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and log in. Many also require a support contract.

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

www.managementsoftware.hp.com/passport-registration.html

Contents

1 Introduction

At the end of this chapter, you will:

- Be familiar with the CM Batch Publisher.
- Understand the different publishing modes available with the CM Batch Publisher.
- Understand the CM Batch Publisher system requirements.

What is the CM Batch Publisher?

The CM Batch Publisher is a command-line-driven content publishing tool that identifies a set of files and components (and their relationships) and publishes them in a controlled, automated, repeatable manner, to the CM-CSDB, where they are stored as objects. The CM Batch Publisher can:

- scan for files on multiple drives or file systems,
- scan and publish files from any mapped drive or file system,
- be configured to limit the subdirectories that are scanned,
- include or exclude at the file level, and
- select files by type.

Also, the CM Batch Publisher can accommodate frequent patching of internal applications, and publish build versions and output from CM legacy (PVCS or ClearCase) adapters. Its capacity to revise content material is reliable, and can be designed to perform continuously, at designated times, and in pre-determined intervals, and can be easily executed from within any script or code capable of calling a command prompt.

Why Use the CM Batch Publisher?

The CM Batch Publisher offers a means of reliable and instant data updates to information that must be posted in an automated fashion.

The primary function of the CM Batch Publisher is to distribute updates to content, data, and applications rather than the initial application packaging. Typically, these types of data updates require a repeatable process. Digital content, such as file sets, graphics, price lists, and interest rates, are types of managed lists that might require an automated update process that the CM Batch Publisher can provide.

Since the CM Batch Publisher is a repeatable process, it dynamically creates package instances and names them (with date and sequence number) to accommodate multiple publishing sessions. The user can select from three input modes: files, input objects, and a configuration file. A CM agent is not required.

The CM Batch Publisher vs. Standard CM Publishing

The CM Batch Publisher provides a command-line alternative to the Component Selection Mode of the graphical user interface of the CM Admin Publisher. The CM Batch Publisher is an automated, repeatable command-line process, whereas the CM Admin Publisher must be monitored from start to finish. For more information on the CM Admin Publisher tool, refer to the *HP OpenView Configuration Management Application Manager Installation and Configuration Guide (CM Application Manager Guide)* or the *HP OpenView Configuration Management Application Self-service Manager Installation and Configuration Guide (CM Software Manager Guide)*.

Support for CM Legacy Adapters

Previous CM Source Control Management Adapters (SCM Adapters) were PVCS and ClearCase (Atria). The CM Batch Publisher is a replacement for these tools, and will accept objects from these legacy adapters.

Overview

The CM Batch Publisher default operation creates standard instances of the PACKAGE, FILE, PATH, DESKTOP, and REGISTRY Classes in the SOFTWARE Domain of the CM-CSDB. Three additional features of the CM Batch Publisher are the ability to:

- publish into other classes, as well as a different domain.
- optionally create (and update, as needed) a ZSERVICE Class instance connection to a published package.
- automatically generate the path information that is required for the distribution of a package. The path information is generated dynamically

by a combination of configuration options and the location of the files being published.

You can run the CM Batch Publisher in one of two ways:

- Provide configuration objects.
- Specify in the configuration file the targeted files to be published.

Table 1 shows how to apply each of these methods.

Table 1 CM Batch Publisher method applications

method	**promote.tkd** (configuration file-based publishing)	**SCMAdapt.tkd** (object-based publishing)
scan	intype=SCAN	ZINPUT.ZAPPLIC=Y
file	intype=FILE (files specified in the insource file)	ZINPUT.ZAPPLIC=N (files specified by heap in ZINPUT or ZPROMOTE)
object	N/A	intype=OBJ
filtering	Available	N/A

Publishing Modes

Configuration File-Based Publishing

Configuration file-based publishing allows for multiple publishing modes that are dictated by the information contained in a configuration file. Multiple configuration files can be maintained and used for different publishing jobs, providing there is an administrator with the ability to repeat a publishing session as needed.

Use the CM Batch Publisher to publish files to the CM-CSDB with either of two methods: scanning a directory or publishing files listed in an input file.

- The scanning method enables you to scan one or more directories. This method also lets you specify:
  - the depth of the scan (that is, the number of subdirectories),
  - filters as selection criteria, and
  - criteria for the inclusion/exclusion of files.
- The files listed method is more efficient if you want to publish a set of files. You can also identify and target files to be published to specific classes of the CM-CSDB. For example, you can designate files with the "lnk" extension to be published to the DESKTOP Class on the CM-CSDB.

In configuration file-based publishing, when a name is designated in the service option and `addtosvc=1`, a new connection is made to the service. If the service does not exist, it is created and the connection is made. In either case, this connection will occupy the first available CONNECT_TO field. In the ZPROMDFT object, which is used in object-based publishing, the ZSERVICE variable must contain a valid instance name, and the ZSVCCNCT variable must be `Y`.

When a name for a package is specified with an asterisk (*), the package name is sequentially generated (prefixYYYYMMDD#) with the same prefix (*prefix**). Multiple packages with the same name (identical prefix*) are linked to one another as REQUIRES connections within the service. The first package promoted is linked directly (as an INCLUDES connection) to the service in the first available CONNECT_TO field. See the following example.

```
SERVICE ---> INCLUDES connection ---> PCKG01
```

Packages (with the same prefix) that are promoted subsequently override the previous package, and assume the direct link to the service, forcing that previous package to adopt a REQUIRES link to it. And so it continues, with each new same-named package breaking its predecessor's INCLUDES connection to the service, and "demoting" that previous package to a REQUIRES link to itself. See the following example.

```
SERVICE--->INCLUDES--->PCKG03
                       |
                       |--->REQUIRES conn--->PCKG02
                                             |
                                             |-->REQUIRES-->PCKG01
```

➤ The prefix used to create a sequentially generated service name must be a unique name and cannot match any existing service names. For example, if the service name SAMPLE exists, the prefix SAMPLE* cannot be used to create sequentially generated service names using the `addtosvc` parameter.

➤ Only in this scenario are the packages connected to the service as REQUIRES, with the second package requiring the first, the third package requiring the second, and so on.

Multiple packages with different names are linked to the service independently at subsequent available connects. Each of these packages will be added in the order in which it is received by the CM Configuration Server, and placed in the first available CONNECT_TO field.

➤ The CM Batch Publisher performs a CRC (cyclical redundancy check) on the fully qualified path, not just the file name. In order for the file to be recognized as a duplicate, it must consistently be promoted from the same location. The CM Batch Publisher does not delete connections, except in the case of multiple promotes having an identical prefix*, nor does it remove REQUIRES links.

Object-Based Publishing

For object-based publishing, the selection of files to be published is derived from information in the ZPROMDFT object *and* either a ZINPUT or ZPROMOTE object, which are generated as a result of the existing CM PVCS and ClearCase adapters.

If you are not using either of these legacy tools, use the CM Admin Agent Explorer to create these objects as described in Input Objects on page 55.

CM Native Packaging

CM Native Packaging is a feature of the CM Batch Publisher specifically designed to publish UNIX native software packages (HP-UX and Solaris). CM Native Packaging is installed with the CM Batch Publisher on UNIX systems. See Chapter 5, CM Native Packaging for more information.

System Requirements and Availability

The CM Batch Publisher is available for Windows and UNIX operating systems. It has these system requirements:

- Network connectivity to the CM Configuration Server.
- A minimum of 2 MB of hard disk space.
- Access to any directories from which you want to publish.

Platform Support

For detailed information about supported platforms, see the release note document that accompanies this release.

Operating System Considerations

Windows Platforms

Registry files that are published into the REGISTRY class need to be converted from the REGEDIT4 registry export format to the CM EDR format required by the CM agent. The CM Batch Publisher will perform this conversion automatically, unless the file has an EDR extension. In this case, `promote.tkd` assumes that the file has already been converted to the EDR format.

⚠ The CM Batch Publisher will *not* convert files from the REGEDIT5 registry export format.

UNIX Platforms

Before using the CM Batch Publisher in a UNIX environment, you need to modify the `filters all` parameter in the configuration file. This is specific to the configuration file-based publishing method (`promote.cfg`).

As you can see below, the default values are:

```
filters all {

type            file
class           file
exclude   "*.log *.bak"
include   "*"
distroot  {}

}
```

You will need to change the `class` parameter from its default of `file` to `unixfile`.

```
filters all {

type            file
class           unixfile
exclude   "*.log *.bak"
include   "*"
distroot  {}

}
```

Make sure that the new class, UNIXFILE, is included in the CM-CSDB. If your CM Configuration Server is version 4.3 or earlier, contact HP Support in order to get the class definition.

The exclude, include, and distroot parameters should be set to the values appropriate to the user's requirements.

Summary

- The CM Batch Publisher is a command-line-driven content publishing tool.
- The CM Batch Publisher offers three publishing modes: Configuration File-Based, Object-Based, and CM Native Packaging.
- The CM Batch Publisher requires connectivity to a CM-CSDB.

2 Installing the CM Batch Publisher

At the end of this chapter, you will:

- Know how to install the CM Batch Publisher.

The CM Batch Publisher is available for Windows and UNIX operating systems. Depending on your operating system, you will need to use either `setup.exe` (for Windows) or `install` (for UNIX) from the installation media for the CM Batch Publisher.

Recommendations

Stop any programs that are running before installing the CM Batch Publisher.

Installing the CM Batch Publisher for Windows

To install the CM Batch Publisher for Windows

1 From the installation media, `/management_extensions/publishing_adapter/publisher/win32` folder, double-click **setup.exe**.

The Welcome window opens.

2 Click **Next**.

The HP Software License Terms window opens.

3 Read the license terms and click **Accept**.

The Directory Location window opens.

4 Type the name of the directory where you would like to install the CM Batch Publisher (default is `C:\Program Files\Hewlett-Packard\CM\BatchPublisher`), or click **Browse** to navigate to it.

5 Click **Next**.

If the directory you specified already exists, you are prompted to replace it.

6 Click **OK**.

   The license file window opens.

7 Enter the location of your license file, or click **Browse** to navigate to it.

8 Click **Next**.

   The Installation Settings window opens.

9 Click **Install**.

10 When the installation is complete, click **Finish**.

You have successfully installed the CM Batch Publisher for Windows.

Installing the CM Batch Publisher for UNIX

If you are installing the CM Batch Publisher on a UNIX system that supports graphics, the graphical installation will automatically begin after you run the installation program. For UNIX systems that support graphics, see UNIX Graphical Installation on page 24. For UNIX systems that do not support graphics, the non-graphical installation program is automatically started. See UNIX Non-Graphical Installation on page 25.

➤ If you are installing the CM Batch Publisher onto a UNIX system that supports graphics, but you would like to use the non-graphical mode instead, change your directory to the location of the install program on the installation media and type:

```
./install -mode text
```

This will start the non-graphical installation of the CM Batch Publisher. See UNIX Non-Graphical Installation on page 25 for instructions.

UNIX Graphical Installation

This section guides you through the graphical installation of the CM Batch Publisher.

To install the CM Batch Publisher using the graphical interface

1. Depending on your version of UNIX, change your current working directory to the correct subdirectory on the installation media.
2. Type **`./install`**, and then press **Enter**.

   The Welcome window opens.
3. Click **Next**.

   The HP Software License Terms window opens.
4. Read the agreement and click **Accept**.

   The Directory Location window opens.
5. Type the name of the directory to which you would like to install the CM Batch Publisher (default is `/opt/HP/CM/BatchPublisher`), or click **Browse** to select a location.
6. Click **Next**.
7. If the directory you specified already exists, you are prompted to overwrite the existing directory. To specify a new directory, click **Cancel** to return to the previous step, or click **OK** to proceed.

   The license file window opens.
8. Enter the location of your license file or click **Browse** to select the location manually.
9. Click **Next**.

   The Installation Settings window opens.
10. Click **Install**.

11 When the installation is complete, click **Finish**.

You have successfully installed the CM Batch Publisher for UNIX.

UNIX Non-Graphical Installation

This section guides you through the non-graphical installation of the CM Batch Publisher for UNIX.

To install the CM Batch Publisher using the non-graphical installation

1 Depending on your version of UNIX, change your current working directory to the correct agent subdirectory on the installation media.

2 Type **`./install -mode text`** and then press **Enter**.

The CM Batch Publisher installation begins.

3 Type **`C`**, and then press **Enter**.

4 Press a key to view the End User License Agreement.

5 When you are finished viewing the agreement, type **`Accept`** and press **Enter**.

6 Accept the default location for the CM Batch Publisher (`/opt/HP/CM/BatchPublisher`) by pressing **Enter**, or specify a different location.

If the directory you specify already exists, you will be prompted to continue. If the directory does not exist, the installation program will display the Installation Settings.

7 Type **`Y`**, and then press **Enter**.

8 Enter the location and name of your license file and press **Enter**.

9 Press **Enter** to accept the default (Y) and begin the installation.

If you do not want to begin the installation, type **`N`**, and then press **Enter**.

10 To complete the configured installation process, press **Enter**.

You have successfully installed the CM Batch Publisher for UNIX.

Summary

- The CM Batch Publisher is available for Windows and UNIX operating systems.
- Before installing the CM Batch Publisher, we recommend that you stop any running programs.

3 Configuration File-Based Publishing (promote.tkd)

At the end of this chapter, you will:

- Be familiar with configuration file-based publishing.
- Understand the command-line parameters needed for `promote.tkd`.
- Understand the `promote.cfg` parameters.
- Understand how to specify additional attributes.

Using Configuration File-Based Publishing

Configuration file-based publishing uses a configuration file (`promote.cfg`) that contains your publishing specifications. The publishing session is then executed from the command line. Command-line parameters are described in Table 2 below, and the configuration file is described in The PROMOTE Configuration File on page 31.

Execute the command line from the directory where you installed the CM Batch Publisher (default is `C:\Program Files\Hewlett-Packard\CM\BatchPublisher\`). The command line is preceded with `nvdkit promote.tkd` files that were installed during the CM Batch Publisher installation and contain the HP runtime Tcl interpreter and configuration file-based publishing code.

Example

```
nvdkit promote.tkd -cfg promote.cfg -user rad_mast -pass
secret
```

Table 2 Command-line parameters for promote.tkd

Parameter	Description
`-cfg` *`filename`*	Specifies the file that contains the configuration options for this execution of the CM Batch Publisher. The file `promote.cfg` is provided as a sample configuration file, and is the default value. This file can be re-named. You can maintain multiple configuration files to facilitate a variety of publishing jobs. This parameter is optional. If no configuration file is specified, `promote.cfg` in the current working directory is used.
`-user` *`userid`*	CM administrator user ID. The default is RAD_MAST. This parameter is optional.
`-pass` *`password`*	CM administrator password. This parameter is optional.

Parameter	**Description**
`-phase input`	If present and the value is `input` (not case-sensitive), the database will be created, but the files will not be published. This is useful for testing filters, debugging, and verifying that your selected criteria are producing the expected results (the results are sent to the log and displayed on the screen). This parameter is optional. Note: Any value other than input will be ignored.

The PROMOTE Configuration File

Table 3 below describes the configuration file parameters.

The PROMOTE Configuration File Format

Table 3 PROMOTE configuration file format (`promote.cfg`)

Option	**Description**
`Package`	Defines the PACKAGE Class instance name or prefix. If specified without a trailing asterisk (*), the value is used as the absolute PACKAGE Class instance name. If specified with a trailing asterisk (*), the value is used as a prefix to dynamically generate the PACKAGE Class instance name. When used as a prefix, the PACKAGE Class instance name is generated as: `<pkgprfx>`*`YYYYMMDDs`* where *`YYYYMMDD`* is the current date, and `s` is a sequence number used to guarantee uniqueness.
`pkgname`	Specifies the friendly name of the PACKAGE Class instance (NAME).
`pkgdesc`	Specifies a description of the PACKAGE Class instance (ZPKGDESC) attribute on the package that gets populated.

Option	Description
`service`	Defines the name of the ZSERVICE Class instance that will be optionally created (or updated) in the CM-CSDB during the publishing session. The publishing session will create a ZSERVICE Class instance if one does not exist. Note: This option will work only if `addtosvc=1`.
`svcname`	Specifies the friendly name of the ZSERVICE Class instance (NAME). This command is optional.
`svcdesc`	Specifies a description of the ZSERVICE Class instance (ZSVCNAME) attribute on the service that gets populated. This command is optional.
`addtosvc`	Tells the CM Batch Publisher whether to update a ZSERVICE Class instance with a connection to the newly published package. `1` = Add connection to ZSERVICE. `0` = Do not add connection to ZSERVICE. Note: If set to `1`, the `service` command must have a value specified.
`compress`	Tells the CM Batch Publisher whether to use compression. `1` = Use compression. `0` = Do not use compression.
`intype`	Defines the type of the input source. Values are `FILE` and `SCAN`. FILE - Use when the list of files to be published is contained in a file. Note: The `insource` option must be used if `intype=FILE`. SCAN - Use when the list of files to be published is to be scanned on a drive/file system. Note: The `filescan` option must be used if `intype=SCAN`.

Option	Description
`insource`	Specifies the name of the source file. The specified file should contain a list of qualified file names, one per line, to be published. Also, `numsplit` and `distroot` can be specified in the file. These options behave in the same manner as described in the filescan row of this table. Note: Relevant only when `intype=FILE`. The formats that are accepted for the lines in the file are: • `global distroot <`*`value`*`>` - Specifies the `distroot` value to be used for the files listed on the lines that follow it. If not specified, the original location of the file will be used as the distribution directory. • `global numsplit <`*`value`*`>` - Specifies the `numsplit` position to be used for the files listed on the lines that follow it. The default value is `1`. • `<filename>` - Specifies the fully qualified name of a file to be published. Notes: Filters will still be applied to the files before publishing. If a file does not match any filters, it will not be published. The commands `global distroot` and `global numsplit` can be specified at any point in the `insource` file. Their values affect only the lines that follow them, and remain in effect until the next `global` command is encountered. Therefore, group together files by their common `distroot` and `numsplit` values. In the examples below, note the values of `numsplit` (`3` and `2`) and `distroot` (`d:/myapps and d:/place`). The resulting outputs are also shown.

Example A **Example B**

```
global numsplit 3              global numsplit 2
global distroot d:/myapps      global distroot
d:/temp/src/apps/a.dat         d:/place
d:/temp/src/apps/test2.tcl     d:/temp/list.pdf
                               d:/temp/mymk.tcl

Output:                        Output:

  (distroot)  |     (stem)       (distroot) |   (stem)
d:/myapps/apps/a.dat           d:/place/list.pdf
d:/myapps/apps/test2.tcl       d:/place/mymk.tcl
```

Option	Description
`mgrdiff`	Reserved for future use. `1` = to activate comparison with existing resources for service. `0` = to turn off.
`loglvl`	Defines the log tracing level. A value of `3` will show informational log messages. A value greater than `3` will show debugging log messages.
`logfile`	Specifies the name of log file.
`host`	Defines the name and port (in URL format) of the host CM Configuration Server. For example: `cmcs://localhost:3464`
`path`	Defines the CM-CSDB path to the file and domain to which the package will be published, for example, `PRIMARY.SOFTWARE.`
`filescan {body}`	Specifies the control information for file scanner. The configuration file sample shows two `filescan` sections, to indicate that multiple `filescan` functions are supported. However, if you are performing only one `filescan` function, you must delete the additional section. Note: This applies only when `intype` is set to `SCAN`. Each `filescan` must contain the following options:
	`dir` - Directory to scan.
	`distroot` - Optional root directory for distribution to be used in the creation of PATH Class instance. If omitted, the root is derived by applying the value of `numsplit` to `dir`.

Option	Description
	`numsplit` - Ordinal position in which to split file paths into root and stem (starting with the drive letter on Win32 systems, and the first directory on UNIX platforms). The root that results from the split will be used in the creation of PATH Class instances, unless `distroot` is specified. The resulting stem is used to create the class instances as specified in the `filters`.{class} option.

Value	Full path	Root	Stem
0	`c:/program files/my app`	`empty`	`c:/program files/my app`
1	`c:/program files/my app`	`c:/`	`program files/my app`
0	`/work/myapp`	`empty`	`/work/myapp`
1	`/work/myapp`	`/work`	`/myapp`

Important Note: We recommend that you specify a minimum value of `1` on Win32 platforms, because a value of `0` will result in the drive letter being included in the stem, rather than the root.

Option	Description
	`depth` - Defines how may directory levels the file scanner will scan, starting with (and including) the directory specified for `dir`. A value of `-1` is a special case that tells the file scanner to scan to any depth. Scan depth cases are:

depth	result
-1	root directory and all of its subdirectories
0	root directory only
1	root directory and its files
>1	root directory and its files down to the specified depth

Option	Description
`filters {body}`	Filters to use as selection criteria during the scan process. Multiple filters are supported. Priority of filters is the order in which they are specified. Therefore, filters for desktop links should be placed before filters for regular files. Once a file meets the selection criteria of a filter, the remaining filters do not evaluate it.

Option	Description
	type - Identifies the type of CM Configuration Server file being filtered. This value tells the publishing session how to create the instance in the CM-CSDB for a given file that matches the filtering criteria. Accepted values are FILE, DESKTOP, and REGISTRY. class – CM-CSDB class used for files selected by filters. For example: FILE, DESKTOP, and REGISTRY. Note: Refer to the section, Operating System Considerations on page 17 for more information. exclude - Specifies a file to be excluded. Values should be enclosed in quotes, with multiple values separated by a space, as in, "*.lnk .exe". This option will accept an asterisk (*) wildcard. include – Specifies a file to be included. Values should be enclosed in quotes, with multiple values separated by a space, as in, "*.lnk *.exe". This option will accept an asterisk (*) wildcard. distroot - Optional root directory (for distribution) to be used in the creation of PATH Class instances for any files that match this filter. Note: This setting overrides the distroot value specified in filescan. value(s) - Optional ZSTOP expression to be used in PACKAGE Class instance. Multiple expressions are supported, and should be arranged as one expression per line.
expression	The ZSTOP expression to be used in the PACKAGE Class instance. Multiple expressions are supported, but should be arranged one per line. This parameter is optional. Note: Although the expression is optional, the variable expression must be specified in the *.cfg file. Its value will be set in ZSTOP in the published package.

Option	Description
`replacepkg`	Replace existing package with new package. This parameter works only for packages that do not contain a PACKAGE connection. If the new package promote session does not complete, the original package remains available renamed with a leading underscore (`_packageName`). If promote session completes successfully, the original package is deleted. `1` = Replace existing package with new package. `0` = Do not replace existing package. If package exists, the CM Batch Publisher session is aborted.

`attr`
`{body}`
	Additional instance attribute values to be added during the promote. The instance names and values should be enclosed in brackets, one per line. Use only valid instance names.
	
	When specifying connection type instances, use an enumerated instance name, with the exception of the first instance, for example, ALWAYS connections should be designated as: `_ALWAYS_`, `_ALWAYS_#2`, `_ALWAYS_#3`. Alternatively, you can specify a connection as `CONN0001`. The enumerated instance names are defined as follows:
	
	METHOD Connections:
	`METH0001, METH0002, METH0003…`
	
	ALWAYS Connections:
	`CONN0001, CONN0002, CONN0003…`
	
	INCLUDES Connections:
	`INCL0001, INCL0002, INCL0003…`
	
	REQUIRES Connections:
	`REQU0001, REQU0002, REQU0003…`
	
	Refer to the section Specifying Additional Attributes on page 40 for more information.

Sample PROMOTE Configuration File

The code below shows a sample `promote.cfg` including the standard defaults. Remove the section in bold if doing only one filescan function.

promote.cfg

```
# CM Batch Publisher Default Options
#
# package      package instance name or prefix (i.e., foo or foo_*)
# pkgname      to be used as friendly name of package (NAME)
# pkgdesc      to be used as description of package (DESCRIPT)
# service      zservice instance name
# svcname      to be used as friendly name of service (ZSVCNAME)
# svcdesc      to be used as a description of the service (NAME)
# addtosvc     connect package to service
# compress     1 to request compression
# intype       source type for list of resources (FILE/SCAN)
# insource     file path for input if type is FILE
# mgrdiff      Reserved for future use.
1 - to activate comparison with existing resources for service
0 – to turn off
#
#
   package     " "
   pkgname     " "
   pkgdesc     " "

   service     " "
   svcname     " "
   svcdesc     " "
   addtosvc    0

   compress    1
   intype      SCAN
   insource    " "

   mgrdiff     0

   loglvl      3
   logfile     promote.log
   host        cmcs://localhost:3464
   path        PRIMARY.SOFTWARE
   replacepkg          1

   #
   # File Scanner Control Info
   # depth     number of subdirs to traverse (-1 = all)
   # numsplit  number of subdirs (includes drive in Win) to use in root
   # distroot  distribution root to be used to create path instance
   #           if left blank, root of dir is used
   #
   filescan {
dir            {}
```

```
distroot              {}
numsplit              2
depth         -1
   }

   filescan {
        dir           {}
        distroot      {}
        numsplit      1
        depth         -1
   }

   #
   # Priority of the component classes as receiving bucket is based on filter order
   #
   # Specialized (like desktop) should be put before file class filters
   #
   # Abstract Filters (multi-type)
   # class     database class used for files that satisfy this filter
   # expressionexpression strings for ZSTOPs in package instance
   #
   filters lnk {
type          desktop
class         desktop
exclude       " "
include       "*.lnk"
distroot              {}
   }

   filters reg {
type          registry
class         registry
exclude       " "
include       "*.reg *.edr"
distroot              {}
   }

   filters all {
type          file
class         file
exclude       "*.log *.bak"
include       "*"
distroot              {}

attr {
        ALWAYS_#3     SOFTWARE.ZSERVICE.REDBOX
        NAME          Redbox
}
```

```
}

expression    {
}
```

Specifying Additional Attributes

Use the CM Batch Publisher `attr` parameter to automatically create Service, Package, and Component instances for individual applications via a publishing session. These additional attribute values can be specified in the configuration file or directly on the command line as command-line arguments.

When specifying additional attributes, the following rules apply:

- The attributes and their values only affect the instances being created or promoted during that publishing session. For example, if the ZRSCVRFY attribute and its value for the UNIXFILE Class are specified as input to the publishing session, only instances of the UNIXFILE Class created during that publishing session are affected. No other instances of the UNIXFILE Class or any other class are affected.
- The value of the attributes, which may share an identical name with attributes in other classes, will not be contaminated by the value specified for a named class. For example, if a CM Batch Publisher execution will create both FILE and UNIXFILE instances in the same publishing session, it is possible to specify an altered value of the ZRSCVRFY attribute for UNIXFILE without altering the default value to be applied to the ZRSCVRFY attribute of the FILE class.
- No new attributes will be added to a class using the CM Batch Publisher. If an additional attribute is specified that is not defined in the class template, the attribute will not be included with the promote object and a warning will be issued in the log file (`promote.log`) as follows:

  ```
  Warning: Invalid Attribute: XYZ!
  Warning: Not defined in class template
  Warning:       -zservice-attr-XYZ discarded
  ```

- Attributes defined in the configuration file will overwrite the attributes inherited from the base instance.
- Attributes defined on the command line will overwrite the attributes defined in the configuration file and the attributes inherited from the base instance.
- The following attributes are generated by the promote process and cannot be specified in the configuration file or on the command line:

  ZRSCDATE

  ZRSCTIME

  ZRSCSIZE

  ZCMPSIZE

  ZRSCSIG

  SIGTYPE

  The following message will be issued to the log if one of these attributes is specified:

  ```
  Warning: Restricted Attribute: ZRSCDATE!
  Warning: ZRSCDATE is set during promote
  Warning:       -all-attr-ZRSCDATE discarded
  ```

- The ZRSCCRC represents a special case. The ZRSCCRC will be calculated if the additional attribute ZRSCCRC is set to YES. Not including the additional attribute will leave the ZRSCCRC field blank.
- There is no error checking of attribute values specified in the configuration file or on the command line. If a value specified is too large for its field or the character type is incorrect, the value will be truncated and the incorrect character type will be promoted. For example, specifying a two-character numeric field such as ZOBJPRI with the value ABCD will result in a value of AB after promotion.

Specifying Additional Attributes in the Configuration File

To specify an additional attribute with its associated value, an `attr` section must be added to the appropriate filter section or class section of the configuration file. Attributes are specified in the filter section for the components they apply to using a unique filter name. Additional Package, Service, and Path attributes are specified in a separate `attr` section.

The sample code below displays an excerpt from a configuration file containing the `all` filter with an additional attribute section (`attr`):

```
filters all {
             type         file
             class        unixfile
             exclude      ""
             include      "*"
             distroot     {/xyz/test}

              attr        {
               ZCREATE    {PKUNZIP &ZRSCCFIL}
               ZPERUID       (&(USER)/&(GRP))
              }
}
```

Within each appropriate filter section an `attr` section is added. The arguments of the `attr` section must be included within curly brackets ({ }). These arguments make up the attribute name and value list for that filter.

The Package, Service, and Path Class instances that are created by the CM Batch Publisher do not have filters associated with them. To specify attributes for these class instances, use the format below, with the attributes and their values specified between the curly brackets ({}).

```
attr PACKAGE {
            RELEASE 3.5.6
}
```

There is only one attribute and its associated value or value list allowed per line. If the value of the variable is multiple words the value must be enclosed in curly brackets ({ }) or double quotes as in the value `{PKUNZIP &ZRSCCFIL}`. Attribute names are not case-sensitive; the values are promoted in the same case in which they are specified.

If an attribute is specified and it is not part of the PACKAGE, ZSERVICE, or PATH Class or it is not part of a recognized filter, the attribute is deleted and the following message is written to the log:

```
Warning: Invalid Filter: abc !
Warning:       -abc-attr-ZUSERID discarded
```

If an attribute specified does not exist in the class template, when this attribute is processed the attribute is discarded and the log will display:

```
Warning: Invalid Attribute: NOTGOOD!
Warning: Not defined in class template
Warning:       -all-attr-NOTGOOD discarded
```

There is no limit to the number of additional attributes that can be specified or the order in which they can be specified.

Specifying Connection Types

INCLUDES, REQUIRES and ALWAYS connections can be specified for all classes that contain these type of connections. There are two methods of specifying connection types.

- Specify the explicit connection type with a sequential number appended such as `_ALWAYS_#3`.
- Specify the numbered type connection such as `CONN0001`.

REGISTRY, DESKTOP, FILE, PACKAGE, and ZSERVICE Classes contain INCLUDES, REQUIRES, and ALWAYS connections defined in the default database. The connection must be specified with the name and the number.

This sample code, displays an example of specifying connections for the ZSERVICE instance.

```
attr zservice {
       _ALWAYS_#3   SOFTWARE.ZSERVICE.REDBOX
       _ALWAYS_#2   SOFTWARE.ZSERVICE.DRAGVIEW
}
```

The connection takes the slot number specified with one exception. The `_ALWAYS_` connection of the ZSERVICE Class is reserved for use by the package instance created by the CM Batch Publisher session. If this connection is specified on the command line or in the configuration file, the value specified in the configuration file or on the command line will overwrite the package connection created from the promote process.

The formats for specifying additional attributes using connection types are as follows:

- Method Connections:
  `METH0001, METH0002, METH0003`
- Always Connections:
  `CONN0001, CONN0002, CONN0003`
- Includes Connections:
  `INCL0001, INCL0002, INCL0003`
- Requires Connections:
  `REQU0001, REQU0002, REQU0003`

The following is an excerpt of the configuration file with the connection type attributes specified.

```
filters all {
        type          file
        class         file
        exclude       "*.log *.bak"
        include       "*"
        distroot      {}
        attr {
   meth0001     notepad
        CONN0003 test123
        }
}
```

A table is printed in the `promote.log` that shows:

- All attributes in the class.
- The connection type (V=variable, M=method, C=class, I=includes, R=requires).
- The connection type name.
- The value inherited from the base instance.
- The value set for the CM Batch Publisher promote.

The following is an excerpt of the table presented in the log file.

```
Info: --------------------------------------------------------
Info:   filter = all   classname = FILE
Info:
Info: Name       Type Connection     BaseInst        RPA
Info: --------------------------------------------------------
Info: ZOBJDATE   V                   20010910
Info: ZOBJTIME   V                   17:04:57
Info: ZOBJID     V                   D0010BE54B1E
Info: ZRSCMO     V                   M               O
Info: ZINIT      M  METH0001                          notepad
Info: _ALWAYS_#3 C  CONN0003                          test123
```

If the same attribute is set using an explicit connection (for example, `ZINIT = {pzunzip &zrsccfil}`) and a connection type connection (for example, `meth0001 = notepad.exe`), the following error is generated and the CM Batch Publisher session is halted.

```
Error:!!!Conflict of Additional Attributes
Error:   Specify either Explicit or Connection type for Attribute
Error:   Explicit type: -all-attr-ZINIT = pzunzip &zrsccfil
Error:   Connection type: -all-attr-METH0001 = notepad.exe
```

Specifying Additional Attributes on the Command Line

Additional attributes can also be specified directly on the command line. Attributes added using the command line take the following format:

```
-(filter name)-attr-(variable name)  value
```

or

```
-(class name )-attr-(variable name)  value
```

Example

```
-all-attr-zinit        "PKUNZIP &ZRSCCFIL"
-package-attr-release        1.2.3
```

Therefore an example of a CM Batch Publisher command line with additional attributes specified would be as follows:

```
nvdkit promote.tkd cfg promote.cfg -all-attr-zinit "PKUNZIP
&ZRSCCFIL"
```

Additional attribute command-line arguments are specified in lowercase with the exception of the attribute values. The attribute values will retain the case they were specified in when promoted. If the value of the attribute contains multiple words, the value should be surrounded by double quotes as in the example above.

The filter name, attr keyword, and variable name must be separated by hyphens.

If the second element of the string is not attr, a warning is issued to the promote.log:

```
Warning: Problem command line attribute !
Warning:      -zservice-axxt-zinit discarded
```

If the configuration file is specified and the `.cfg` file exists, no new configuration file is unpacked. If the configuration file does not exist, a blank configuration file is unpacked with the name specified for the `.cfg` file. If no `.cfg` file is specified, the default name of `promote.cfg` is used for the blank configuration file that is unpacked.

When the `promote.tkd` is run, a sample `.cfg` file is unpacked.

Filters and Filescans

To specify filters and filescan configuration on the command-line use the following formats.

Filescans

Only one filescan can be specified on the command line. If additional filescans are needed they must be specified in the configuration file. The command-line options for filescan are:

```
-fs-dir
-fs- distroot       {}
-fs- numsplit       1
-fs- depth          -1
```

Filters

To specify a filter on the command line use the following argument format:

```
-filters <filtername>
-<filtername>-type        value
-<filtername>-class       value
-<filtername>-exclude     value
-<filtername>-include     value
```

You must use the filters argument to specify the unique name of the filter. There can be multiple filter entries each specifying a unique filter name. Multiple filters can be defined on the command line.

Command-line example:

```
nvdkit promote.tkd -filters testrpa -testrpa-type file -
testrpa-class file -testrpa-exclude "" -testrpa-include "*"
```

The filter executed on the command line above is displayed in the `promote.log` excerpt below:

```
20020918 11:42:05 Info: Filter[testrpa]:
20020918 11:42:05 Info: filtername = testrpa
20020918 11:42:05 Info:       type = file
20020918 11:42:05 Info:      class = file
20020918 11:42:05 Info:    include = *
20020918 11:42:05 Info:    exclude = {}
```

There is no limit to the number of additional attributes that can be specified or the order in which they can be specified. The same rules that apply to the configuration file for valid attributes also apply to the command-line attributes.

Specifying attributes on the command line, the attribute must be in a recognized filter or in the zservice, package, or path class. If not, the following message is written to the log:

```
Warning: Invalid Filter: abc !
Warning:       -abc-attr-ZUSERID discarded
```

If a package name is not specified on the command line, the default package name of `rpadefault*` is used.

```
#
# package        - package instance name or prefix (i.e. foo or foo_*)
# pkgname       - to be used as friendly name of package (NAME)
# pkgdesc       - to be used as description of package (DESCRIPT)
# service       - zservice instance name
# svcname       - to be used as friendly name of the service (ZSVCNAME)
# svcdesc       - to be used as a description of the service (NAME)
# addtosvc      - connect package to service
# compress      - 1 to request compression
# intype- source type for list of resources (FILE/SCAN)
# insource      - file path for input if type is FILE
# mgrdiff       - 1 to activate comparison with existing resources for service - not implemented
#
#
    package     "attr_test"
    pkgname     "attr_test"
    pkgdesc     "attr_test"

    service     "attr_test"
    svcname     "attr_test"
    svcdesc     "attr_test"
    addtosvc    1

    compress    1
    intype      SCAN
    insource    ""

    mgrdiff     0

    loglvl      3
    logfile     promote.log
    hostcmcs://localhost:3464
    pathPRIMARY.SOFTWARE
#
    # File Scanner Control Info
    # depth     - number of subdirs to traverse (-1 = all)
    # numsplit  - number of subdirs (includes drive in win) to use in root
    # distroot  - distribution root to be used to create path instance
    #             if left blank, root of dir is used
    #
    filescan {
        dir             {c:/attr/test}
        distroot        {}
        numsplit        2
        depth           2
    }
    #
    # Priority of the component classes as receiving bucket is based on
    # filter order
    # Specialized (like desktop) should be put before file class filters
    #
    # Abstract Filters (multi-type)
    # class     - database class used for files that satisfy this filter
    # expression - expression strings for ZSTOPs in package instance
    #
filters reg {
            type         registry
            class        registry
            exclude      ""
            include      "*.reg *.edr"
            distroot     {}
}

filters lnk {
        type         desktop
        class        desktop
        exclude      ""
        include      "*.lnk"
        distroot     {}
        attr    {
        MACHUSER    TESTUSER
        ZCREATE     {PKUNZIP &ZRSCCFIL}
        }
    }
filters all {
                type         file
                class        file
                exclude      ""
                include      "*"
                distroot     {/john/test}
                attr    {
                ZCREATE     TESTSTART
                ZDELETE     TESTOVER
                }
    }
  expression {
    }
attr package {
            releASE     3.5.6
            wrong       thisiswrong
            includes    SOFTWARE.PACKAGE.ADAPT
            includes#2  SOFTWARE.PACKAGE.RAPILINK
        }
        attr  zservice {
        ZSVCMO          m
        URL             {WWW.HP.COM}
        _ALWAYS_#3      SOFTWARE.ZSERVICE.REDBOX
        _ALWAYS_#2      SOFTWARE.ZSERVICE.DRAGVIEW
        }
     attr path {
```

```
                    zrscmo  0
}
```

Summary

- Execute configuration file-based publishing from the command line.
- Edit `promote.cfg` to include your required publishing parameters.
- Use the `attr` parameter to specify additional attributes.

4 Object-Based Publishing (SCMAdapt.tkd)

At the end of this chapter, you will:

- Be familiar with object-based publishing.
- Understand `SCMAdapt.tkd` command-line parameters.
- Understand the `SCMAdapt.cfg` parameters.

Using Object-Based Publishing

SCMAdapt.tkd is a file that is included with the CM Batch Publisher installation media. When you use it for object-based publishing, the CM Batch Publisher takes the input or output objects from one of the CM legacy Source Control Management adapters (EDMPVCS or EDMATRIA) and publishes the files to CM. Use the command-line arguments discussed in Table 4 below, to do this. The configuration file parameters are described in The SCMADAPT Configuration File on page 59.

Execute `SCMAdapt.tkd` on a command line from the directory where you installed the CM Batch Publisher (default is `C:\Program Files\Hewlett-Pacakard\CM\BatchPublisher`). Once executed, `SCMADapt.tkd` uses the supplied arguments to determine the location of the objects and configuration file for the publishing session.

Example

```
nvdkit scmadapt.tkd -objdir <Object Directory> -cfg
<scmadapt.cfg>-user <userid> -pass <password> -phase input
```

Table 4 Command-line parameters for scmadapt.tkd

Parameter	Description
`-objdir` *`object directory`*	If a valid set of objects is not found, `SCMAdapt` will terminate. This parameter is required.
`-cfg` *`filename`*	Specifies the file that contains the configuration options for this execution of the CM Batch Publisher. This parameter is optional. If not present, the `scmadapt.cfg` file in the current working directory will be used. If `scmadapt.cfg` is not found, `SCMAdapt` will terminate. This file can be re-named. You can maintain multiple configuration files to facilitate a variety of publishing jobs. This parameter is required. See Table 6 on page 60 for a description of the configuration file parameters.
`-user` *`userid`*	CM administrator user ID. The default is `RAD_MAST`. This parameter is optional.

Parameter	Description
`-pass` *`password`*	CM administrator password. The default is " " (no password). This parameter is optional.
`-phase input`	Optional parameter. If present and the value is `input` (not case-sensitive), the database will be created, but the files will not be published. This is useful for testing filters, debugging, and verifying that your selected criteria are producing the expected results (the results are sent to the log and displayed on the screen). Note: Any value other than `input` will be ignored.

Input Objects

SCMAdapt requires an input of two objects, ZPROMDFT, and one of the others described below. All of these objects are from a CM legacy SCM Adapter.

- ZPROMDFT object
  default values for the adapter.

and

- ZINPUT object
  input to the adapter. (Use the CM Admin Screen Painter or the CM Admin Agent Explorer to build input to `SCMAdapt`.)

or

- ZPROMDFT object
  default values for the adapter.

and

- ZPROMOTE object
  output of the adapter (output from a CM legacy SCM Adapter).

If the secondary input object is ZINPUT, no SCM access is done. Only the file or application defined in the heap will be published.

ZPROMDFT Variables

Table 5 below shows the variables of the ZPROMDFT object. Although all the variables listed might appear in a CM legacy SCM adapter object, those marked N/A are not used by the CM Batch Publisher. Using an object editor (such as the CM Admin Agent Explorer), open the ZPROMDFT object to ensure that the required variables are present.

Table 5 ZPROMDFT Variables

Variable	Description
UNIQUE	N/A
ZADMCLAS	N/A
ZADMDOMN	N/A
ZADMFILE	N/A
ZADMIPRE	N/A
ZADMMLOC	N/A
ZAPPNAME	N/A
ZCOMPRESS	N/A
ZEXETYPE	N/A
ZPACKAGE	The instance name of the package. This variable is required. If this variable has an asterisk (*) suffix, the value will be used as a prefix. The instance names will contain this prefix, followed by the date and a sequence number.
ZPKGDESC	A description of the package. This variable is required.
ZPKGNAME	The friendly name of the package. This variable is required.
ZPROMDIR	N/A
ZPROMOTE	N/A

Variable	Description
ZSERVICE	Contains the instance name of the service to which the published package should be attached. Note: If the instance name does not conform to instance naming rules, an error will be generated.
ZSVCCNCT	Defines whether a service should be connected. If `Y`, attach (connect) the package to the service. This requires that ZSERVICE and ZSVCNAME be present for the service to be connected. If `N`, do not attach the package to the service.
ZSVCNAME	The friendly name of the service. This variable must exist if ZSERVICE exists.
ZTRACEL	N/A
ZVLBLTYP	N/A
ZSERVICE	Contains the instance name of the service to which the published package should be attached. Note: If the instance name does not conform to instance naming rules, an error will be generated.
ZSVCCNCT	Defines whether a service should be connected. If `Y`, attach (connect) the package to the service. This requires that ZSERVICE and ZSVCNAME be present for the service to be connected. If `N`, do not attach the package to the service.
ZSVCNAME	The friendly name of the service. This variable must exist if ZSERVICE exists.
ZTRACEL	N/A
ZVLBLTYP	N/A

ZSERVICE, ZSVCNAME, and ZSVCCNCT are not *required* variables.

However, they all must be present in order for a package to be connected to a service. If any of these variables is missing, a notice is sent to the log and the console, and the package will be created and the resources published, but the package will not be connected to a service.

ZINPUT Variables

These variables have to be added to the ZINPUT object:

ZSPLIT

The position in the ZPRPCFIL value where to split the file name into a root and a stem.

The value of the root becomes the PATH instance value, and the FILE instance value adopts the value of the stem. For example:

```
ZPRPCFIL: F:\intstage\s054ptest\test\bin\TESTING.TXT ZSPLIT: 2
```

The root is: `F:\intstage\`.

The stem is: `s054ptest\test\bin\TESTING.TXT`

A PATH instance is created with a value of: `F:\intstage\`.

A FILE instance is created with a value of: `s054ptest\test\bin\TESTING.TXT`.

ZDSTROOT

The deployment location.

If present, the PATH instance will be set to this value.

If not present, the PATH instance will be set to the root directory of the promoted resource (for example, `&(ZRSCCDRV)&(ZRSCCDIR)`).

The ZDSTROOT value will be retrieved from the ZINPUT object, if it is present.

If ZDSTROOT is not specified, the path from where the resource was published will be assumed.

ZCLASS

This variable identifies the file type specified in the heap. The acceptable values are FILE, DESKTOP, and REGISTRY. The default is FILE.

If ZCLASS is not present, a message indicating that the variable is being defaulted will be printed in the log and on the console.

ZCLASS is only honored on ZINPUT heaps that specify a file, not an application (such as, when ZAPPLIC=N).

ZAPPLIC

The flag that states whether the ZINPUT heap is an application.

If Y, ZPRPCFIL will be the base directory (root) from where to start the file scan, and all its files and subdirectories will be published.

If N, ZPRPCFIL will be the file to be published.

The SCMADAPT Configuration File

The following two sections present contain information on the SCMADAPT configuration file.

The SCMADAPT Configuration File

This section contains a table in the format of the configuration file.

Table 6 **SCMADAPT configuration file parameters**

Option	Description
`compress`	Tells the CM Batch Publisher whether to use compression. `1` = Use compression. `0` = Do not use compression.
`intype`	Defines the type of the input source. `OBJ` is the only valid value.
`mgrdiff`	Reserved for future use. `1` = to activate comparison with existing resources for service. `0` = to turn off.
`loglvl`	Defines the log tracing level. A value of `3` will show informational log messages. A value greater than `3` will show debugging log messages.
`logfile`	Specifies the name of log file.
`host`	Defines the name and port (in URL format) of the host CM Configuration Server, for example, `cmcs://localhost:3464.`
`path`	Defines the CM-CSDB path to the file and domain to which the package will be published, for example, `PRIMARY.SOFTWARE.`
`fileclass`	Sets the file class name. This command is specific to SCMAdapt, and will override the defaults of FILE (Win32) and UNIXFILE (UNIX).
`expression`	The ZSTOP expression to be used in the PACKAGE Class instance. Multiple expressions are supported, but should be arranged one per line. This parameter is optional. Although the expression is optional, the variable expression must be specified in the `*.cfg` file. Its value will be set in ZSTOP in the published package.

The SCMADAPT Configuration File Sample

This section contains a sample `scmadapt.cfg` that shows the standard defaults.

It should not be necessary to modify the default configuration file (except for the *host* value) unless the name of the `fileclass` is to be changed.

```
scmadapt.cfg

#
# compress  1 to request compression
# intype    source type for list of resources (OBJ)
# insource  This field must be present, but must not be specified
# mgrdiff   Reserved for future use.
            1 - to activate comparison with existing resources
for service
            0 – to turn off
# fileclass File class name - defaults to FILE in Win Platforms
#                             defaults to UNIXFILE on UNIX
platforms
    compress 1
    intype   OBJ
    insource " "

    mgrdiff  0

    loglvl    3
    logfile   SCMAdapt.log
    host      cmcs://localhost:3464
    path      PRIMARY.SOFTWARE
    fileclass " "
    expression {
    }
```

Summary

- Execute object-based publishing from the command line.
- Edit `SCMAdapt.cfg` to include your required publishing parameters.

5 CM Native Packaging

At the end of this chapter, you will:

- Be familiar with CM Native Packaging.
- Understand CM Native Packaging system requirements.
- Understand the CM Native Packaging command-line interface.
- Know how to publish using CM Native Packaging.

What is CM Native Packaging?

CM Native Packaging is a feature of the CM Batch Publisher specifically designed for UNIX environments. It is a command-line-driven content-publishing tool that:

- Supports native UNIX software.
- Is neither a graphical publishing tool nor a mainstream publishing tool.
- Is installed during the regular installation of the CM Batch Publisher on a UNIX system.
- Explores UNIX native software depots and searches for available native packages
- Publishes wrapped native packages to the CM Configuration Server. It will publish all necessary information that will allow you immediate installation of native software to end clients including, if necessary, information about native package dependencies.

Why use CM Native Packaging?

CM Native Packaging supports HP-UX (SD), Solaris (SVR4, patches, patch clusters and rpm), AIX (bff, rte, and rpm), and RedHat Linux RPM software package formats. With the use of CM Native Packaging you can easily publish wrapped native UNIX software, updates, and patches without any need for re-packaging. Wrapped UNIX native software enables policy-based centralized software management of your UNIX agents.

This document assumes that the system administrator who uses the CM Native Packager possesses packaging/publishing knowledge for a CM Configuration Server DB.

Overview

CM Native Packaging creates the standard instances of ZSERVICE, PACKAGE, and PATH in the SOFTWARE Domain of the CM-CSDB. CM Native Packaging creates instances of SD, SVR4, SOLPATCH, AIX, or RPM classes for each published wrapped native package depending on the operating system (HP-UX, Solaris, AIX or RedHat Linux).

For each native software package selected, CM Native Packaging will create an instance of SD, SVR4, SOLPATCH, AIX, or RPM class. This instance holds actual content (software depot) and native method calls that will do actual install/removal/update on the client. It will also create an instance of the PACKAGE Class that contains the newly created instance and an instance of ZSERVICE Class that contains the new PACKAGE instance.

Publish native packages from the specific UNIX platform to which you will be deploying. For example, you cannot use CM Native Packaging on HP to promote Solaris SVR4 packages and or Solaris Patches – CM Native Packaging would be unable to use the native UNIX utilities to interrogate details of the package.

CM Native Packaging System Requirements

CM Native Packaging is available for the HP-UX, Solaris, AIX, and RedHat Linux operating systems. It has these system requirements:

- Root permissions are required to use CM Native Packaging.
- Network connectivity to the CM Configuration Server.
- Space on `/tmp` file system for temporary depot files used for publishing.

Response files are only supported with Solaris SVR4 native software packages.

Required Classes

CM Native Packaging requires specific classes for each operating system. Make sure your CM-CSDB includes these SOFTWARE Domain classes before using CM Native Packaging.

Table 7 Required SOFTWARE Domain Classes

Operating System	Class
Solaris	SVR4 Packages (SVR4)
	SVR4 Package Datastreams (SVR4)
	Solaris Patches (SOLPATCH)
	Solaris Patch Clusters (SOLPATCH)
	Solaris RPM Packages (RPM)
HP-UX	SD Packages (SD)
	SD Product Bundles (SD)
AIX	IBM AIX Packages (AIX)
	IBM RPM Packages (RPM)
RedHat Linux	Linux RPM Packages (RPM)

CM Native Packaging and the CM Agent

During the installation of the CM agent, a Tcl script is installed into the `IDMSYS` directory along with the CM agent components. This script is required for deployment of packages published using CM Native Packaging. The actual Tcl script installed depends on your UNIX operating system. The scripts (`sd.tcl` for HP-UX, `svr4.tcl`, `rpm.tcl`, and `solpatch.tcl` for Solaris, `aix.tcl` and `rpm.tcl` for AIX, `rpm.tcl` for RedHat Linux) contain native command calls to deploy the software.

A common helper Tcl script `method_utils.tcl` is also installed with the CM agent, on all platforms where CM Native Packaging is supported.

Supported Native Package Types

Table 8 below lists the native package types supported by CM Native Packaging and their expected formats.

Table 8 Native Package and Supported Formats

Native Package	Supported Format
HP-UX SD Product	File system format (extracted to disk, the software depot contains subdirectories reflecting the SD Product tag as well as the SD depot catalog).
HP-UX SD Product/Patch Bundle	File system format (extracted to disk, the software depot contains subdirectories reflecting the SD Product tag as well as the SD depot catalog).
SVR4 Package	File system format (software depot contains a subdirectory reflecting the Package, e.g., `SUNWdtpcv`, and optional response file, e.g., `SUNWdtpcv.response`).
SVR4 Package Datastream	`*.pkg` file and optional respective `*.response` file Packages require a .pkg extension. If the package does not have `.pkg` extension, the extension should be added.
Solaris Patch	File system format (software depot contains a subdirectory reflecting the Patch number, e.g., 111111-03)
Solaris Patch Cluster	File system format (software depot contains a subdirectory reflecting the Patch Cluster, e.g., `8_Recommended`).
Solaris RPM Package	`*.rpm` file
AIX Bff Package	`*.bff` file

Native Package	Supported Format
AIX Rte Package	*.rte file
AIX RPM Package	*.rpm file
RedHat™ Linux RPM Package	*.rpm file

CM Native Packaging Command-Line Interface

CM Native Packaging is run from the command line. The base input parameter for CM Native Packaging is the source depot containing HP-UX, Solaris, AIX, or RedHat Linux software. The native packages must be in a disk depot format (the native software packages are resident on disk in a format that can be utilized immediately by the native operating system's software management tools). CM Native Packaging is capable of publishing one or more packages in a single publishing session.

In addition, you can specify the selection of the software you want to publish, and in the event CM-CSDB user verification is enabled, an optional user ID and password can be designated. Here is an example of command-line usage for CM Native Packaging:

```
20070201 18:18:40 Info: Message catalog for en_us loaded.
Usage: rnp -d depot_path -m manager_ip:manager_port
    [-v] [-debug type] [-tmp directory]
    [-user user_id] [-pass password] [-admin]
    [-depth level] [-dist dist_depot_path]
    [-domain domain] [-l logfile] [-help]
    [-i] [-coreq] [-I] [-M] [-S]
    [-a | -A type | -p
package1[,r=revision][,a=architecture][,v=vendor]
                -p
package2[,r=revision][,a=architecture][,v=vendor]...]
    [-P] [-r] [-f prefix]
    [-s] [-t service_type] [-c flag] [-relyondb]
```

The table below contains the description of the command-line arguments for CM Native Packaging.

Table 9 Command-line parameters

Parameter	Description
`-a`	Specifies to publish all native software available in the depot. This parameter is optional. You cannot use this parameter with `-p`.

`-A` *`type`*	Select and publish all packages of specific *`type`*.
	
	`type` can also be one of the following:
	
	`help`
	for a list of valid types for the running platform.
	
	`all`
	to select all package types. This option would then behave like the `-a` option.
	
	`none`
	to select none of the package types. This would then behave like having neither the `-a` or `-A` options specified.
	
	Multiple package types can be specified and separated by commas.
	
	This parameter is optional.

`-c flag`	This option enables or disables compression on all packages to be published. *`flag`* can be one of the following: • `yes` Enable compression for all packages. • `no` Disable compression for all packages. Default behavior is dependent on each package type being published. This parameter is optional.
`-coreq`	Includes co-requisite (on HP-UX) packages promoted into a "mini-depot."
`-d depot` *`path`*	Specifies the path to the depot directory containing native software packages. Software contained in this depot will serve as an input to CM Native Packaging. This parameter is required.

Parameter	Description
`-debug` *`type`*	Specify the level of debugging desired. *`type`* can be one of the following: `init` for initialization data `func` for detailed function debugging `trace` for function tracing `cmd` for native command executions `pub` for publishing information `rapi` for CM Batch Publisher details `all` for all the above `none` to disable debugging Multiple types can be specified and separated by commas. The default behavior is that debugging is disabled. This parameter is optional.
`-depth`	Linux RPM packages only. Determines the level of dependency processing for the target package. 0 – Process all dependencies. 1 – (Default) process 1 level of dependency. You can specify any number of level dependency you require. If no level is defined (-depth is not part of the rnp command), then one level of dependencies is processed (assuming –i option used). This option allows for CM Batch Publisher backwards compatibility.
`-dist distribution_depot`	Linux RPM packages only. Specifies the path to a distribution depot directory. Published packages will contain DIST=distribution_depot in the CONTENTS field of their package class instance.

Parameter	Description
`-domain` *`domain`*	Specify which CM Configuration Server domain the packages are to be published to. The default domain used is PRIMARY.SOFTWARE. This parameter is optional.
`-f prefix`	Instructs CM Native Packaging to prefix the PACKAGE Class and service class instance names used for the new published package with this prefix. This parameter is optional.
`-help`	Display help on the command-line usage and the `rnp.cfg` configuration file format.
`-i`	Instructs CM Native Packaging to include prerequisite software package (supported for SD, SVR4, Solaris patches and RPM packages only) with the package you have selected if prerequisite software is present in the source depot. Dependency information is published regardless of this parameter. This parameter is optional.
`-I`	Interactive mode. Allows user to select more required packages (dependency). Ignored if neither –i nor –coreq are present or no additional package is required. Note: Available for SD, SVR4, Solaris patches and RPM packages only.
`-l` *`logfile`*	Instructs CM Native Packaging to store the log details in the *`logfile`* specified. If this option is omitted, the default log file created is `publish.log`. This parameter is optional.
`-m` *`ip:port`*	Specifies the host name or IP address and port of the CM Configuration Server to which you intend to publish software. This parameter is required.

Parameter	Description
-M	Multiple. If -i or -coreq is present (so additional packages are required), and there are several versions of an additional package, then all of them will be displayed in the additional packages menu. Otherwise, only one version of each additional package will be displayed (default). It is ignored if -I is not present.
-p *package* [,r=revision] [,a=arch] [,v=vendor]	Specifies a software package to publish to the CM Configuration Server. Specify the following: an SVR4 package or Solaris Patch (when specified with the -P option) on Solaris, a bff package on AIX, an RPM package on RedHat Linux, An SD product software selection on HP-UX (software selection with optional revision, architecture and vendor. Specifying the software selection alone will work, but if there are multiple products with the same identifier, they will all be published). This parameter is optional. You can specify multiple -p *package* parameters for multiple package selections. On HP-UX the following parameters can be used to define more specific package selection: r = Revision number of software being published a = Architecture v = Operating System Vendor Note: If a package is not specified on the command line, you will be presented with a list of all available packages within the specified depot.
-P	Instructs CM Native Packaging to publish Solaris Patches instead of SVR4 packages on SunOS. This parameter is optional and available only for Solaris.
-pass *password*	CM administrator password. This parameter is optional.

Parameter	Description
-r	Instructs CM Native Packaging to publish a response file for Solaris non-interactive installation support. A <*pkgname*>.response file that must reside in depot_path is promoted together with the Solaris package file. If the response file does not exist the package is promoted without it. Available only on Solaris systems.
-s	Instructs CM Native Packaging to skip the creation of services for the packages to be published.
-S	Strict mode. If any requirements for a package are not met (for example, if -i or -coreq option are present and not all additionally required packages are in the depot), the package will not be promoted. It is ignored if -I option is present.
-t *svc_type*	Use this option to specify the type of service to create. Available values: M for Mandatory O for Optional Default Service type created is M. This parameter is ignored when the -s option is specified.
-tmp *dir*	Specify an alternative temporary directory to use when creating packages. The default value is /tmp. This parameter is useful when */tmp* on the machine where publishing is performed has limited available disk space. This parameter is optional.
-user *user ID*	CM administrator user ID. The default is RAD_MAST. This parameter is optional.
-v	Displays the version and build number of the CM Native Packager rnp command. This parameter is optional.

When no packages are specified with the -p option or by selecting all packages with the -a or -A options, the CM Native Packaging command will present a text based menu of native packages found in the depot directory specified. You can then select individual or all packages from the menu to be published.

CM Native Packaging Options File

If you usually use the same source depot, or publish to the same CM Configuration Server, you can create a file, rnp.cfg, in the same directory where you have the CM Native Packaging components installed. Use of this configuration file allows you to preset default option values in the following format:

parameter=value

Example:

```
depot=<depot path>
manager_ip=<CM configuration server IP or hostname>
manager_port=<port number that the CM configuration server
uses>
```

By default, rnp.cfg is not supplied.

Table 10 Supported rnp.cfg settings and default values

Setting	Expected Values	Default Value
depot	Fully qualified path to the depot directory	None
manager_ip	IP address or hostname of the CM Configuration Server	None
manager_port	Port number of the CM Configuration Server	manager_port=3464
user	user=userid Administrator ID used for authentication with the CM Configuration Server.	User=RAD_MAST

Setting	Expected Values	Default Value
`create_service`	`create_service=[yes/no]` A value of `yes` will create a ZSERVICE instance for each of the promoted packages. A value of `no` will not automatically create a ZSERVICE instance for each of the promoted packages	`create_service=yes`
`service_type`	`service_type=[M/O]` A value of `M` will cause the promoted ZSERVICE instance to be set as a mandatory service. A value of `O` will cause the promoted ZSERVICE instance to be set as an optional service.	`service_type=M`
`select_patches`	`select_patches=[yes/no]` A value of `yes` will set the default publishing behavior on Solaris to be for the publishing of patches. Value of `no` will set the default behavior on Solaris to be for the publishing of SVR4 packages.	`select_patches=no`
`include_responses`	`include_responses=[yes/no]` A setting of `yes` will include SVR4 response files when they are found in the Solaris depot. Value of `no` will not include response files for Solaris SVR4 packages.	`include_responses=no`

Setting	Expected Values	Default Value
`include_dependencies`	`include_dependencies=[yes/no]` A value of `yes` will attempt to publish SVR4, RPM, or SD dependent packages if they are in the specified depot. A value of `no` will not attempt to publish SVR4, RPM, or SD dependent packages.	`include_dependencies=no`
`include_corequisites`	`include_corequisites=[yes/no]` A value of `yes` will attempt to publish SVR4 or SD dependent packages if they are in the specified depot. A value of `no` will not attempt to publish SVR4 or SD dependent packages.	`include_corequisites=no`
`include_adminfile`	`include_adminfile=[yes/no]`	`include_adminfile=no`
`select_types`	`select_types=[type1,type2,…]` Publish all packages of specific types found in the depot directory. Run rnp with the `-A` help option to get a complete list of supported types for the running platform.	`select_types=none`
`debug`	`debug=[type1,type2,…]` List specific types of debugging to enable. valid types are: init, func, trace, cmd, pub, rapi, all or none.	`debug=none`

Setting	Expected Values	Default Value
`temp_dir`	`temp_dir=[dir]` Specify an alternate temporary directory to use for creating the packages to publish.	`temp_dir=/tmp`
`domain`	`domain=FILE.DOMAIN` Specify the target FILE.DOMAIN in the CM-CSDB where to publish the packages.	`domain=PRIMARY.SOFTWARE`
`compress`	`compress=[yes/no]` Enable or disable compression for all packages to be published. The default behavior is that compression depends on the package type being published.	`Package Dependent`
`password`	`password=pass` Administrator password, used for authentication with the CM Configuration Server.	`Blank`
`interactive`	`interactive=[yes/no]` Publish using interactive mode. Interactive mode allows you to choose whether or not to include required packages.	`interactive=no`
`strict`	`strict=[yes/no]` Publish using strict mode. Strict mode will not publish packages missing required components.	`strict=no`
`multiple`	`multiple=[yes/no]` Display multiple versions of additional required packages,	`multiple=no`
`rely_on_db`	`rely_on_db=[yes/no]`	`rely_on_db=no`
`requisite_depth`	`Set to 0 to include all dependencies or set to a number of levels.`	`requisite_depth=1`
`distribution_depot`	`Distribution depot path.`	`Blank`

Publishing with CM Native Packaging

Examples

See Table 9 on page 69 for an explanation of the CM Native Packager command-line parameters.

To publish SD product SD_PROD from default depot on HP-UX

1 Change your current working directory to the CM Batch Publisher directory.

2 On the command line, type:

```
./rnp -user rad_mast -pass secret -d /var/spool/sw -p
SD_PROD,r=1.0,v=HP
```

To publish all SVR4 packages residing in the default depot on Solaris

1 Change your current working directory to the CM Batch Publisher directory.

2 On the command line, type:

```
./rnp -d /var/spool/pkg -a
```

Or, if the depot contains packages as well as patches or patch clusters, type the following:

```
./rnp -d /var/spool/pkg -A pkg
```

To publish a specific Solaris Patch residing in the specified depot

1 Change your current working directory to the CM Batch Publisher directory.

2 On the command line, type:

```
./rnp -d /var/spool/patch -p 111111-03 -P
```

To publish a specific Solaris Patch Cluster residing in the specified depot

1. Change your current working directory to the CM Batch Publisher directory.
2. On the command line, type:

   ```
   ./rnp -d /var/spool/patch -p 8_Recommended -P
   ```

To publish a fileset in package `dce.client.core.rte-4.3.0.0.bff` residing in the `/usr/sys/inst.images` depot on AIX

1. Change your current working directory to the CM Batch Publisher directory.
2. On the command line, type:

   ```
   ./rnp -d /usr/sys/inst.images -p dce.client.core.rte-
   4.3.0.0.bff
   ```

To publish a specific Xchat RPM package residing in the specified depot on RedHat Linux

1. Change your current working directory to the CM Batch Publisher directory.
2. On the command line, type:

   ```
   ./rnp -d /home/rpmadmin -p xchat-1.4.0.2.i386.rpm
   ```

   Or simply:

   ```
   ./rnp -d /home/rpmadmin -p xchat
   ```

   If a package is not supplied on the command line via the `-p` parameter, you will be presented with a list of all available packages within the specified depot.

Publishing with Interactive Mode

When you specify the parameter`-I` on the command line, the CM Native Packager interactive mode is invoked. This allows you to select which of the *available* required software you would like to include with your current package. You will also see which required prerequisite software is not available in the current depot.

The interactive mode option is ignored if neither the `-I` nor `-coreq` or `-i` parameters are specified on the command line (indicating prerequisite software is required for the current package). Here is an example of Interactive Mode:

```
---------------------------------------------------------
Processing additional software required for QA_MASTER_1-1.0.0-0.i386.rpm

Following additionally required software is found in software depot
and selected to be included in to promote package:
      1. prereqs:  - QA_RPM2-1.2.0-0.i386.rpm - included
      2. prereqs:  - QA_RPM3-1.0.0-0.i386.rpm - included
      3. prereqs:  - QA_RPM4-1.0.0-0.i386.rpm - included

Please toggle the selection:
Select (a to include all; d to exclude all; c to continue; s to skip current package; q to
quit entire session; a number to toggle its selection):
```

You can exclude any of the required software by entering the corresponding number. A message at the end of each line (included or not included) lets you know whether or not the required software will be included with the current package.

- Enter the number of the required software or type another option available in the interactive mode menu and press **Enter** to continue the native packaging process.

Table 11 Interactive Mode Selections

Selection	Description
a	Selects all available required software to include with the current package. Available required software is included by default. (Set all available required software to included)

Selection	Description
c	Continue the native packaging process. If you have made changes to the list of packages to be processed, the continue option will update this list. Use the continue option again to move on to the next processing step. This behavior accommodates multi-level dependency processing.
d	Deselects all included software. (Set all available required software to not included)
q	Quit the CM Native Packaging process.
s	Skip the current package.

Wrapped Native Packages

The following section lists all CM-CSDB class instances and their attributes that are created when you publish native UNIX software with CM Native Packaging.

CM Native Packaging utilizes a **method harness** to invoke agent methods, therefore when a package is published to the CM-CSDB, populated method attributes such as ZCREATE, ZDELETE, ZUPDATE, ZVERIFY, and ZREPAIR will contain the text "hide nvdkit method."

The supplied agent methods are designed to invoke the native software management utilities, therefore, the methods are not interchangeable between agent platforms. For example: the file `sd.tcl` supplied with HP-UX CM agents invokes native HP-UX package management utilities and therefore the successful execution of this method on an operating system other than HP-UX is not possible.

An exception exists when the native software management utilities are available on multiple platforms. The RPM package type and related native utility rpm is supported and available under RedHat Linux as well as AIX and Solaris. The `rpm.tcl` file for the RPM client methods is thus supplied with these platforms.

When publishing native UNIX packages using the CM Native Publisher, the software packages are published to the CM-CSDB (in compressed format). The depot containing native software in compressed format is promoted to SD class (class is similar to UNIXFILE Class). The tables below list the modified attributes:

Table 12 SD Class instance attributes modified

Attribute	Description
ZRSCNAME	Specifies a string that is used by native methods to identify software contained in the published depot. This is the complete software spec on HP-UX (tag, version, architecture and vendor).
ZRSCCFIL	Specifies the path to the file that is included in this instance. This file contains the native packaged software.
AUTOBOOT	This Boolean variable is set to Y in case the wrapped SD software contains a reboot file set.
ZCREATE	Uses method Harness call. The CM agent method `sd.tcl` script contains a native command call to install the software package: `hide nvdkit method` Note: If all the file systems listed in `/etc/fstab` are not mounted, ZCREATE (swinstall) will fail. This default behavior assures that later installations will work correctly.
ZDELETE	Uses method Harness call. The CM agent method `sd.tcl` script contains a native command call to remove the software package: `hide nvdkit method` Note: When a native software application is removed, the application files are deleted, but the directory structure will remain.
ZUPDATE	Uses method Harness call. The CM agent method `sd.tcl` script contains a native command call to update the software package: `hide nvdkit method`
ZVERIFY	Uses method Harness call. The CM agent method `sd.tcl` script contains a native command call to verify the installed software package: `hide nvdkit method`

Attribute	Description
ZREPAIR	Uses method Harness call. The CM agent method `sd.tcl` script contains a native command call to repair the installed software package(reinstall): `hide nvdkit method`
ADDDEPS	Auto-select dependencies. Set to N by default.
PREREQ	Software specification of prerequisite SD product. Note that SD's dependencies are on the fileset level. Since CM Native Packaging wraps SD products, dependencies are elevated to product level. Informational attribute only.
COREQ	Software specification of corequisite SD product. Informational attribute only.
EXREQ	Software specification of exrequisite SD product. Informational attribute only.
CONTENTS	Required Packages Included in Tar. Note: If the promoted package is a bundle, CONTENTS will contain the value BUNDLE, and the attributes PREREQ, COREQ and EXREQ will contain no value.
INSTOPTS	Package installation options. (For example, -xenforce_dependencies=true -xallow_downdate=true) NOCHECK keyword can be added to installation options. See Operational Notes for more information.

Table 13 SVR4 Class Instance Attributes Modified by CM Native Packaging

Attribute	Description
ZRSCNAME	Specifies a string that is used by native methods to identify software contained in the published depot. This is the SVR4 package name.
ZRSCCFIL	Specifies the path to the file that is included in this instance. This file contains the native packaged software.
ADMIN	Admin File Exists? [Y/N]

Attribute	**Description**
ADMINOBJ	Is this admin object? [Y/N]
AUTOBOOT	This Boolean variable is set to Y. In the event you do not want the native SVR4 package management to trigger a reboot (exit codes 10 or 20 of pkgadd/pkgrm), manually set this attribute to N.
CONTENTS	Required Packages Included in Tar
ZCREATE	Uses method "Harness" call. The CM agent method svr4.tcl script contains a native command call to install the software package: `hide nvdkit method`
ZDELETE	Uses method "Harness" call. The CM agent method svr4.tcl script contains a native command call to remove the software package: `hide nvdkit method`
ZUPDATE	Uses method "Harness" call. The CM agent method svr4.tcl script contains a native command call to update the software package: `hide nvdkit method`
ZVERIFY	Uses method "Harness" call. The CM agent method svr4.tcl script contains a native command call to verify the installed software package: `hide nvdkit method`
ZREPAIR	Uses method "Harness" call. The CM agent method svr4.tcl script contains a native command call to repair the installed software package (reinstall): `hide nvdkit method`
RESPONSE	Specifies if a response file was published with the package (Y/N). Default value is N.
RESPOBJ	Specifies whether this instance is the response file (Y/N). Default value is N.
RESPFILE	Path to response file (if it exists). Default is empty string.
PKGVER	Package Version
PKGREV	Package Revision

Attribute	Description
PREREQ	Name of prerequisite SVR4 package. Informational attribute only. See Note below
INCOMP	Name of incompatible SVR4 package. Informational attribute only. See Note below
REVERSE	Name of SVR4 package with reverse dependency to base package. Informational attribute only. See Note below
INSTOPTS	Package installation options. NOCHECK keyword can be added to installation options. See Operational Notes for more information.

The SVR4 dependency attributes PREREQ, INCOMP, and REVERSE are only populated when dependency information exists for SVR4 packages in disk depot format only and not when the packages are already bundled as datastreams (`.pkg` files).

Table 14 SOLPATCH Class Instance Attributes

Attribute	Description
ZRSCNAME	Specifies a string that is used by native methods to identify software contained in the published depot. This is the Solaris Patch Identifier.
ZRSCCFIL	Specifies the path to the file that is included in this instance. This file contains the native packaged software.
CLUSTER	Specifies whether this instance pertains to a Solaris patch cluster (Y/N). Default value is N.
ZCREATE	Uses method "Harness" call. The CM agent method solpatch.tcl script contains a native command call to install the software patch: `hide nvdkit method`

Attribute	Description
ZDELETE	Uses method "Harness" call. The CM agent method solpatch.tcl script contains a native command call to remove the software patch: `hide nvdkit method`
ZUPDATE	Uses method "Harness" call. The CM agent method solpatch.tcl script contains a native command call to update the software patch: `hide nvdkit method`
ZVERIFY	Uses method "Harness" call. The CM agent method solpatch.tcl script contains a native command call to verify the installed software patch: `hide nvdkit method`
ZREPAIR	Uses method "Harness" call. The CM agent method solpatch.tcl script contains a native command call to repair the installed software patch (reinstall): `hide nvdkit method`
CONTENTS	Required packages included In Tar

Table 15 AIX Class Instance Attributes

Attribute	Description
ZRSCNAME	Specifies a string that is used by native methods to identify software contained in the published depot. This is the package filename without the bff or rte suffix on AIX (all dots are converted to underscores).
ZRSCCFIL	Specifies the path to the file that is included in this instance. This file contains the native packaged software.
ZCREATE	Uses method "Harness" call. The CM agent method aix.tcl script contains a native command call to install the software package: `hide nvdkit method`
ZDELETE	Uses method "Harness" call. The CM agent method aix.tcl script contains a native command call to remove the software package: `hide nvdkit method`

Attribute	Description
ZUPDATE	Uses method "Harness" call. The CM agent method aix.tcl script contains a native command call to update the software package: `hide nvdkit method`
ZVERIFY	Uses method "Harness" call. The CM agent method aix.tcl script contains a native command call to verify the installed software package: `hide nvdkit method`
ZREPAIR	Uses method "Harness" call. The CM agent method aix.tcl script contains a native command call to repair the installed software package (reinstall): `hide nvdkit method`
COMMIT	If turned on, this flag commits all file sets in package update. Default value is N.
FORCE	If turned on this flag forces the installation of a software product even if (there exists) a previously installed version of the software product that is the same as or newer than the version currently being installed. Default value is N.
PREREQ	Name of prerequisite AIX file sets. Informational attribute only.
COREQ	Name of co requisite AIX file sets. Informational attribute only.
INSTREQ	Name of installed requisite AIX file set. Informational attribute only.
IFREQ	Name of if requisite AIX file set. Informational attribute only.

Table 16 RPM Class Instance Attributes

Attribute	Description
ZRSCNAME	Specifies a string that is used by native methods to identify software contained in the published depot. This is the RPM Package Name.

Attribute	Description
ZRSCCFIL	Specifies the path to the file that is included in this instance. This file contains the native packaged software.
ZCREATE	Uses method "Harness" call. The CM agent method `rpm.tcl` script contains a native command call to install the software package: `hide nvdkit method`
ZDELETE	Uses method "Harness" call. The CM agent method `rpm.tcl` script contains a native command call to remove the software package: `hide nvdkit method`
ZUPDATE	Uses method "Harness" call. The CM agent method `rpm.tcl` script contains a native command call to update the software package: `hide nvdkit method`
ZVERIFY	Uses method "Harness" call. The CM agent method `rpm.tcl` script contains a native command call to verify the installed software package: `hide nvdkit method`
ZREPAIR	Uses method "Harness" call. The CM agent method `rpm.tcl` script contains a native command call to repair the installed software package (reinstall): `hide nvdkit method`
PKGVER	Package Version. Informational attribute only.
PKGREL	Package Release. Informational attribute only.
PKGARCH	Package Architecture. Informational attribute only.
PKGSUMM	Package Summary. Informational attribute only.
REQPKGS	Required Packages. Informational attribute only.
REQCMDS	Required Commands. Informational attribute only.
REQLIBS	Required shared libraries. Informational attribute only.
CONTENTS	Required packages included In Tar
PKGEPOCH	RPM Package EPOCH.

Attribute	Description
INSTOPTS	Package installation options. (For example, --oldpackage , --replacepkgs). NOCHECK keyword can be added to installation options. See Operational Notes for more information.
VRFYOPTS	Package verify options. (For example, --nomode , --nogroup)

An instance of PACKAGE class is created that contains the instance of SD,SVR4,SOLPATCH,AIX, or RPM class. Table 17 below describes how CM Native Packaging maps native package information into PACKAGE class attributes.

Table 17 PACKAGE Class attributes

Attribute	Description
Instance Name	On HP-UX CM Native Packaging will take SD product tag, prefix `SD_` and append a date and sequence number to guarantee uniqueness (`SD_<tag>_`*`yyyymmddn`*). On Solaris only the `SVR4_` string is pre-pended to SVR4 package name and a date and sequence number is appended to the name to guarantee uniqueness (`SVR4_<PKG>_yyyymmddn`). For Solaris Patches, the `SOLPATCH_string` followed by the patch OS is pre-pended to the Solaris Patch number (`SOLPATCH_5_8_<PATCH>`). For Solaris Patch Clusters, the SOLPATCH_ string is pre-pended to the Patch Cluster Identifier and a date and sequence number is appended to guarantee uniqueness (SOLPATCH_<PATCH_CLUSTER>_yyyymmddn). For AIX bff and rte packages, the `AIX_` prefix is added to package name and date and sequence number is appended (`AIX_<PKG>_yyyymmddn`). For RPM, the `RPM_` prefix is added to the RPM Package Name and date and sequence number is appended (`RPM_<PKG>_yyyymmddn`). Note: When instance names generated are longer than 32 characters, the package/patch names parts of the instance

Attribute	Description
	names shall be truncated.
RELEASE	SD revision, SVR4 and RPM version native attributes are mapped into RELEASE. For AIX bff and rte packages, no release information is mapped since it is usually included in package name (filename). For Solaris Patches, the patch revision is mapped into RELEASE. For Solaris Patch Clusters, no release information is mapped since it is usually reflected in the friendly name.
NAME	On HP-UX, NAME is composed from SD_ and SD product's software spec (`SD_<`*`software_spec`*`>`). On Solaris NAME is the same as instance name • `(SVR4_<PKG>_yyyymmddn, SOLPATCH_5_8_<PATCH>` or • `SOLPATCH_<PATCH_CLUSTER>`). On AIX, AIX_ prefix is added to AIX package name without the bff or rte suffix. For RPM Packages, RPM_ prefix is added to the RPM package name and suffixed with the package version , release and architecture `(RPM_<PKG>,ver=<VERSION>,rel=<RELEASE>,arch=<ARCH>).`
DESCRIPT	SD's title and SVR4's name attributes are mapped into DESCRIPT. For SOLPATCH, the Patch or Patch Cluster synopsis is mapped into DESCRIPT. On AIX description of the package is mapped in DESCRIPT. If package description doesn't exist, description of the first fileset in the package is mapped. For RPM Packages, the package summary is mapped into DESCRIPT.
ZSTOP000	Contains an expression that contains target operating system information.

Attribute	Description
ZSTOP001	On HP-UX possibly contains SD products target OS release.
FILE	Holds reference to respective instance of `SD/SVR4/SOLPATCH/AIX/RPM` class.

CM Native Packaging also creates an instance of ZSERVICE Class holding previously created instance of PACKAGE Class. Table 18 on page 89 lists the modified attributes.

Table 18 ZSERVICE Class attributes

Attribute	Description
Instance Name	On HP-UX, the CM Native Packager will take the SD product or bundle tag, prefix SD_ and append a date and sequence number to guarantee uniqueness (`SD_<TAG>_yyyymmddn`). For Solaris SVR4 Packages, the SVR4_ string is pre-pended to the package name and a date and sequence number is appended to the name to guarantee uniqueness (`SVR4_<PKG>_yyyymmddn`). For Solaris Patches, the `SOLPATCH_string` followed by the patch OS is pre-pended to the Solaris Patch number (`SOLPATCH_5_8_<PATCH>`). For Solaris Patch Clusters, the SOLPATCH_ string is pre-pended to the Patch Cluster Identifier and a date and sequence number is appended to guarantee uniqueness (`SOLPATCH_<PATCH_CLUSTER>_yyyymmddn`). For AIX bff and rte packages, the AIX_ prefix is added to the package name and a date and sequence number is appended to the name (`AIX_<PKG>_yyyymmddn`). For RPM, the RPM_ prefix is added to the RPM Package Name and a date and sequence number is appended (`RPM_<PKG>_yyyymmddn`). Note: When instance names generated are longer than 32 characters, the package/patch names parts of the instance names shall be truncated.

Attribute	Description
VERSION	SD revision or SVR4 version native attributes are mapped into VERSION. For Solaris Patches, the patch revision is mapped into VERSION. For AIX bff or rte packages, no release information is mapped since it is usually included in the package name (filename). For RPM Packages, the RPM Package Version is mapped into VERSION.
NAME	On HP-UX NAME is composed from SD_ and SD product's software spec (`SD_<software_spec>`). On Solaris NAME is the same as instance name (`SVR4_<PKG>`). For Solaris Patches and Patch Clusters, the name is composed of SOLPATCH_ followed by the Patch or Patch Cluster Synopsis. On AIX, the AIX_ prefix is added to the AIX package name without the bff or rte suffix. For RPM Packages, the RPM_ prefix is added to the RPM package name and suffixed with the package version, release and architecture (`RPM_<PKG>,ver=<VERSION>, rel=<RELEASE>,arch=<ARCH>`).
ZSVCNAME	SD's title, SVR4, or SOLPATCH name attributes are mapped into ZSVCNAME. On AIX, AIX_ prefix is added to the AIX package name without the bff or rte suffix. For RPM, the RPM Package Name is mapped into ZSVCNAME.
VENDOR	Specifies vendor of the native UNIX package. Not applicable on AIX.
ZSVCMO	Service is set to mandatory by default. Valid values of this attribute are: • `M` for mandatory • `O` for optional
ALWAYS	Holds reference to the respective instance of PACKAGE Class.

If a package requires a system reboot after an agent connect, make sure the `hreboot radskman` parameter is set to `Y`. Refer to the CM *Application Manager Guide* for more information.

Automatic Inclusion of Required Packages

If you specify the `-i` command-line option, CM Native Packaging will include prerequisite packages into the depot with the (main) package you are publishing to CM. The prerequisite package needs to exist in the depot CM Native Packaging is using as a source. (This feature is not supported for AIX (bff) packages, Solaris Patches and Patch Clusters) For RPM packages, all prerequisite packages as well as any additional prerequisite packages they might require are included (this can be determined by using the –depth option). All other package formats include only the first level of required packages for the package to be published.

The `-coreq` command option will include corequisite packages for SD (COREQ).

When using the `-i` or `-coreq` options, the promotion of native software packages will not fail because of a missing prerequisite or corequisite package (unless the `-S` option is specified). Installation will fail only if prerequisite or corequisite packages are missing from both the promoted native software package *and* from the target machine.

Alternatively, if a prerequisite or corequisite package is already installed on the target machine, including these in a native software package for promotion will result only in using more network bandwidth and disk space than necessary.

Publishing packages with prerequisites included may take an extended amount of time. About every thirty seconds, a progress message is displayed:

```
Info:    Compiling extended info about all packages in the depot.
Please wait...
```

This feature is not supported for AIX (bff) packages, Solaris Patches and Patch Clusters.

Troubleshooting CM Native Packaging

Should you encounter problems publishing native UNIX software packages, please perform the following steps before contacting technical support:

- Enable full diagnostic tracing by appending the text `-debug all` to your command line and re-run the publishing session.
- Have the log file produced by the CM Native Packager publishing readily accessible to provide to support. By default, the log file would be called `publish.log` located in the directory where you installed the CM Batch Publisher.

➤ You should only use the command-line option `-debug all` to diagnose publishing problems.

Operational Notes

The following describes the operations involved during the publishing and deployment of native packages. This gives you a better understanding of the current processes and capabilities provided to manage these packages

Publishing

- All packages are selected from the software depot specified by using the -d option. We recommend that you build a depot with packages of the same architecture only (for example, separate depots of RPMs for installation of i386 or x86_64 machines).
- "-dist *a_distribution_depot*" option (RPM packages only) where *a_distribution_depot* is the location of packages used during deployment. If this option is present, then the -d option can be omitted and *a_distribution_depot* location will be used instead. Packages promoted with the -dist option will contain "DIST=distribution_depot_path" in the CONTENTS field of their package class instances.

- Dependency checking is performed on the target (selected) package or patch as well as on all its dependent packages (currently multi-level dependency checking is implemented for RPM packages only). Use "–depth N" option to control the depth of dependency processing. If the –depth option is not defined, only one level of dependencies (of the target package only) are processed. To include all dependencies, use –depth 0.

- Dependencies are not checked when processing certain package formats that contain multiple entries (such as HP-UX bundles and Solaris data streams). This process is typically performed when these 'bundles' are created.

- On Linux, a dependency's release level can be specified as conditional (>= version 2, release 1). If multiple dependencies are found to satisfy this condition, by default the newest package is selected for inclusion. If a specific version is desired, one can use the –M option in interactive mode to list all possible matches, and select the one desired.

- (Solaris only) To specify an installation defaults file ("admin file"), use the –admin option. The name of this file should equal the name of the target package, and have an extension of ".admin". This file must be located in the same software depot where the package is published from. When this file is included in the deployment, it is used specifically for that deployment. If this option is not used, the global admin file ("admin.svr4") located in the IDMSYS directory will be used.

- Use the `-S` (strict) option to ensure that all identified dependencies are included in the deployment. If required dependencies are not found in the software depot, an error message will be displayed and publishing will be terminated.

- On AIX, dependency checking is for informational use only, and the dependencies are not currently included in the deployment.

- SVR4 packages promoted from a depot in datastream format must be first renamed with the `.pkg` extension.

- Using Interactive mode allows you to:

  — See all packages in the software depot available for selection

  — Review all dependencies found for a selected package or patch

— Select / de-select dependencies. This allows administrators who have knowledge of their target machines to tailor the deployment to fit their environments and needs. Some dependencies can be large, and rather than waste bandwidth and client processing, if not needed, it can be removed from the deployment.

Deployment

- If a package was promoted with the –dist option, it will be installed from the distribution depot specified in the CONTENTS field.
- If the target package is already installed on the machine and is newer than the one to be deployed, no further processing is done, and the deployment is viewed as successful. However, since it was not deployed, it will not be removed when the service is deleted.

  ➤ If back leveling of the package is required, this behavior can be overridden by specifying the appropriate native command-line option in the attribute INSTOPTS for SD and RPM packages. This requires the use of the CM-CSDB Editor

  ➤ For SVR4 packages, this behavior can be overridden by adding the NOCHECK keyword to INSTOPTS attribute of the package. This requires the use of the CM-CSDB Editor.

- If (during installation) the target package already exists and is the same release level, it is first verified. If verification fails, it will be re-installed. Subsequent verify or delete processing would occur as usual. This behavior can be changed by adding NOCHECK keyword into INSTOPTS attribute of the package (this requires the use of the CM-CSDB Editor). If present, the package will be re-installed even if it does not fail verification. It is valid for SD, SVR4 and RPM packages only. Appropriate options for the installation are still required.
- (Solaris and Linux only) During install/update, the release levels of already installed dependencies are individually checked, and if newer on machine, they are not installed as this may cause conflicts for other packages. This behavior can be changed by adding the NOCHECK keyword to the INSTOPTS attribute of the package (this requires the use of the CM-CSDB Editor). For example, if INSTOPTS for an RPM package

is set to “NOCHECK --oldpackage” or “NOCHECK --force”, then required packages for the package can be back leveled; while without NOCHECK, only the main package can be back leveled. For SVR4 packages, the presence of NOCHECK in INSTOPTS is sufficient For HP-UX, this process is performed during the analysis phase of SWINSTALL execution.

- During verify, only the target package is verified and not its dependencies.
- After installation, the native package database is queried to make sure the target package was properly installed and registered in the database.
- When installing an HP-UX (SD) patch, the method will first check to see if it has been superseded. If so, no further processing is performed, as it is regarded as obsolete.
- During removal, the package is checked to make sure it exists (as it may have been upgraded or superseded). If it does not exist, no attempt to delete it is made, and the process is viewed as successful. Only the target package is deleted. Dependent packages are not deleted, as they may be required for other packages.
- If the verify attribute (ZRSCVRFY) of the package instance is set to N, the source depot (file actually deployed from server) is deleted after a successful installation. If a subsequent verification of the installed target package fails, this file is again downloaded and used to repair the damaged package.

Event Reporting

Use the RNPEVENT object to report events to the CM Configuration Server. Similar to the APPEVENT object, RNPEVENT uses the same variable set and is created if the CM administrator has enabled the reporting flags for a particular event in the EVENTS variable of the ZSERVICE Class. The RNPEVENT variables are listed in the table below.

Table 19 RNPEVENT variables

Variable	Description	Sample Value
EVENT	Text description of the current event.	create
STATUS	Error messages.	Successful
CMDRC	Return code from native command.	0
CMDMSG	Message from native command.	Main package <Regina> on desktop <2.0> is newer than in CM-CS <1.0>. Skipping further processing.
POSTRC	Return code from RNP post-processing check.	0
POSTMSG	Message from RNP post-processing check	Post installation is successful
ZOBJDOMN	The domain name for the application.	SOFTWARE
ZOBJCLAS	The class name for the application.	ZSERVICE
ZOBJNAME	The instance name for the application.	RPM_GAIM_200504123
ZOBJID	The objects ID for the instance.	D123ACD45F67
ZUSERID	CM user that installed the application.	RPMUSER_LINUX
DELDATE	ISO8601 date time when the delete event occurred.	2005-05-10T16:45:00Z
VERDATE	ISO8601 date time when the verify event occurred.	2005-06-10T16:47:00Z
INSTDATE	ISO8601 date time when the install event occurred.	2005-07-10T16:44:00Z
FIXDATE	ISO8601 date time when the repair event occurred.	2005-08-10T16:43:00Z

Variable	Description	Sample Value
UPGDATE	ISO8601 date time when the update event occurred.	2005-09-10T16:42:00Z
JOBID	Session identifier	MachineConnect
CJOBID	Session identifier	11122:3

Viewing Event Details

Use the CM Reporting Server to view the details of your CM Native Package Events. View the details of the CM Managed Service, then select the CM Native Package Events report. Refer to the *CM Reporting Server Guide* for details on using the CM Reporting Server.

Summary

- CM Native Packaging is a feature of the CM Batch Publisher specifically designed for UNIX environments.
- CM Native Packaging requires specific classes for each operating system.

A Product Name Changes

If you have used Radia in the past, and are not yet familiar with the newly rebranded HP terms and product names, Table 20 below will help you identify naming changes that have been applied to the Radia brand.

Table 20 Product Name and Term Changes

New Name/Term	Old Name/Term
HP OpenView Configuration Management Batch Publisher	Radia Publishing Adapter
HP OpenView Configuration Management Administrator	Radia Administrator Workstation
HP OpenView Configuration Management	Radia
HP OpenView Configuration Management Admin Agent Explorer	Radia Client Explorer
HP OpenView Configuration Management Admin CSDB Editor	Radia System Explorer
HP OpenView Configuration Management Application Manager	Radia Application Manager,
HP OpenView Configuration Management Application Self-service Manager	Radia Software Manager
HP OpenView Configuration Management Configuration Server	Radia Configuration Server
HP OpenView Configuration Management Configuration Server Database	Configuration Server Database, Radia Database
HP OpenView Configuration Management Inventory Manager	Radia Inventory Manager

Index