HP OpenView Publishing Adapter Using Radia

Radia Publishing Adapter Guide

Software Version: 4.0

for the UNIX and Windows operating systems

Manufacturing Part Number: T3424-90073

August 2004

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

Trademark Notices

Linux is a registered trademark of Linus Torvalds.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Acknowledgements

Technical Support

Please select Support & Services from the following web site:
<http://www.hp.com/managementsoftware/services>

There you will find contact information and details about the products, services, and support that HP OpenView offers.

The support site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

Preface

About this Guide

Who this Guide is for

This guide should be used by Radia systems administrators who possess knowledge of publishing and packaging for a Radia infrastructure database.

What this Guide is about

The *Radia Publishing Adapter Guide* is an introduction and installation guide for the Radia Publishing Adapter for Windows and UNIX platforms. It covers the two Radia Publishing Adapter modes of publishing: Configuration File-Based Publishing and Object-Based Publishing. In addition, the *Radia Publishing Adapter Guide* covers Radia Native Packaging.

Summary of Changes

This printing of the *Radia Publishing Adapter Guide* for use with Version **4.0** contains the following changes to information and procedures for the following chapters. This version of the second printing contains some organizational changes and modifications to enhance understanding and ease of use.

3.0.1 Note

3.0.1 Items with this symbol represent changes that are specific to version **3.0.1**.

Chapter 1: Introduction

- Page 19: Added a note about the use of prefixes in service names.

Chapter 3: Configuration File-Based Publishing

- Page 44, *Table 3.2*: added the `replacepkg` variable. It defines the PACKAGE class instance name or prefix.
- Page 44, *Table 3.2*: added the `attr` variable. It represents additional instance attribute values to be added during the promote.
- Page 47, *Specifying Additional Attributes*: new section that describes how to use the `attr` variable.

Chapter 5: Radia Native Packaging

3.0.1 Information about AIX, SOLPATCH, RedHat Linux, and RPM was added throughout this chapter.

- Page 71, *Overview*: Added a note about publishing native packages.

3.0.1 Page 71, *Required Classes*: new section that includes a new table: *Table 5.1 ~ Prerequisite Classes*.

- Page 72, *Figure 5.1 ~ Command-Line usage for Radia* Native Packaging: new figure.

Editorial Improvements

In addition to the changes listed above, this version contains various editorial and style updates to each chapter and section and the index.

Conventions

You should be aware of the following conventions used in this book.

Table P.1 ~ Styles

Element	Style	Example
References	*Italic*	See the *Publishing Applications and Content* chapter in this book.
Dialog boxes and windows	**Bold**	The **Radia System Explorer Security Information** dialog box opens.
Code	`Andale Mono`	`radia_am.exe`
Selections	**Bold**	Click **Next** to continue.

Table P.2 ~ Usage

Element	Style	Example
Drives (system, mapped, CD)	Italicized placeholder	*SystemDrive*:\Program Files\Novadigm might refer to C:\Program Files\Novadigm on your computer. *CDDrive*:\client\radia_am.exe might refer to D:\client\radia_am.exe on your computer.
Files (in the Radia Database)	All uppercase	PRIMARY
Domains (in the Radia Database)	All uppercase	PRIMARY.SOFTWARE May also be referred to as the SOFTWARE domain in the PRIMARY file.
Classes (in the Radia Database)	All uppercase	PRIMARY.SOFTWARE.ZSERVICE May also be referred to as the ZSERVICE class in the SOFTWARE domain in the PRIMARY file.

The table below describes terms that may be used interchangeably throughout this book.

Table P.3 ~ Terminology*

* Depends on the context. May not always be able to substitute.

Term	May also be called
Application	software, service
Client	Radia Application Manager and/or Radia Software Manager
Computer	workstation, server
NOVADIGM domain	PRDMAINT domain **Note**: As of the 4.0 release of the database, the NOVADIGM domain is being renamed the PRDMAINT domain. Therefore, if you are using an earlier version, you will see the NOVADIGM domain in the database.
Radia Configuration Server	Manager, Active Component Server
Radia Configuration Server Database	Radia Database

Contents

1

Introduction

At the end of this chapter, you will:

- Be familiar with the Radia Publishing Adapter.
- Understand the different publishing modes available with the Radia Publishing Adapter.
- Understand the Radia Publishing Adapter system requirements.

What is the Radia Publishing Adapter?

The Radia Publishing Adapter 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 Radia Database, where they are stored as objects. The Radia Publishing Adapter can:

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

Additionally, the Radia Publishing Adapter can accommodate frequent patching of internal applications, as well as publish build versions, and output from Novadigm 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 Radia Publishing Adapter?

The Radia Publishing Adapter offers a means of reliable and instant data updates to information that must be posted in an automated fashion.

The primary function of the Radia Publishing Adapter 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 Radia Publishing Adapter can provide.

Since the Radia Publishing Adapter 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 Radia Client is not required.

The Radia Publishing Adapter vs. Standard Radia Publishing

The Radia Publishing Adapter provides a command-line alternative to the Component Selection Mode of the graphical user interface of the Radia Publisher. The Radia Publishing Adapter offers an automated, repeatable command-line process, whereas the Radia Publisher must be monitored

from start to finish. For more information on the Radia Publisher tool, refer to the *Radia Application Manager Guide* or *Radia Software Manager Guide.*

Support for Radia Legacy Adapters

Previous Radia *Source Control Management Adapters* (SCM Adapters) were PVCS and ClearCase (Atria). The Radia Publishing Adapter is intended as a replacement for these tools, and will accept objects from these legacy adapters.

Overview

The Radia Publishing Adapter default operation creates standard instances of the PACKAGE, FILE, PATH, DESKTOP, and REGISTRY classes in the SOFTWARE domain of the Radia Database. Three additional features of the Radia Publishing Adapter 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.

The Radia Publishing Adapter is run one of two ways:

- By providing configuration objects.
- By specifying in the configuration file the targeted files to be published.

Table 1.1 ~ Radia Publishing Adapter Method Applications below shows how to apply each of these methods.

Table 1.1 ~ Radia Publishing Adapter Method Applications

	promote.tkd	SCMAdapt.tkd
	configuration file-based publishing	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, which are dictated by the information contained in a configuration file. Multiple configuration files can be maintained and used for different publishing jobs, providing an administrator with the ability to repeat a publishing session as needed.

Files can be published to the Radia Database using either method available with the Radia Publishing Adapter, *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. Additionally, you can identify and target files to be published to specific classes of the Radia Database. For example, you can designate files with the "lnk" extension to be published to the DESKTOP class on the Radia Database.

  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 doesn't 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, 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
  ```

  Subsequent packages (with the same prefix) that are promoted 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
```

Note

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.

Note

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 Radia Configuration Server, and placed in the first available CONNECT_TO field.

Note

The Radia Publishing Adapter 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 Radia Publishing Adapter 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 *and* either a ZINPUT or ZPROMOTE object, which are generated as a result of the existing Radia PVCS and ClearCase adapters.

If you are not using either of these legacy tools, use the Radia Client Explorer to create these objects as described in *Input Objects* on page 61.

Radia Native Packaging

Radia Native Packaging is a feature of the Radia Publishing Adapter specifically designed to publish UNIX native software packages (HP-UX and Solaris). Radia Native Packaging is installed with the Radia Publishing Adapter on UNIX systems. See *Radia Native Packaging* starting on page 69 for more information.

System Requirements and Availability

The Radia Publishing Adapter is available for Win32 and the AIX, HP-UX, Linux, and Solaris UNIX operating systems. It has these system requirements:

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

Operating System Considerations

Win32 Platforms

Registry files being published into the REGISTRY class need to be converted from the REGEDIT4 registry export format to the Radia EDR format required by the Radia Client. The Radia Publishing Adapter 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.

> **Caution**
>
> The Radia Publishing Adapter will *not* convert files from the REGEDIT5 registry export format.

UNIX Platforms

Before using the Radia Publishing Adapter in a UNIX environment, it is necessary to modify the `filters all` parameter in the configuration file (see the sample PROMOTE configuration file in Table 3.3 on page 45). This consideration is specific to the configuration file-based publishing method (`promote.cfg`).

As you can see in Table 3.3 on page 46, 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      {}

}
```

Important Notes

Make sure that the new class, UNIXFILE, is included in the Radia Database. If your Radia Configuration Server is version 4.3 or earlier, contact Novadigm Technical 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. Refer to the `filters` row, of Table 3.2 on page 41 for more information on these parameters.

Summary

- The Radia Publishing Adapter is a command-line driven content publishing tool.
- The Radia Publishing Adapter offers three publishing modes: Configuration File-Based, Object-Based, and Radia Native Packaging.
- The Radia Publishing Adapter requires connectivity to a Radia Database.

2

Radia Publishing Adapter Installation

At the end of this chapter, you will:

- Know how to install the Radia Publishing Adapter.

The Radia Publishing Adapter 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 CD media to install the Radia Publishing Adapter.

Recommendations

Stop any programs that are currently running before installing the Radia Publishing Adapter.

Installing the Radia Publishing Adapter for Windows

To install the Radia Publishing Adapter for Windows

1. From the installation media, double-click **Setup.exe**.

   The **Welcome** window opens.

Figure 2.1 ~ Welcome window of the Radia Publishing Adapter.

2. Click **Next**.

The **Directory Location** window opens.

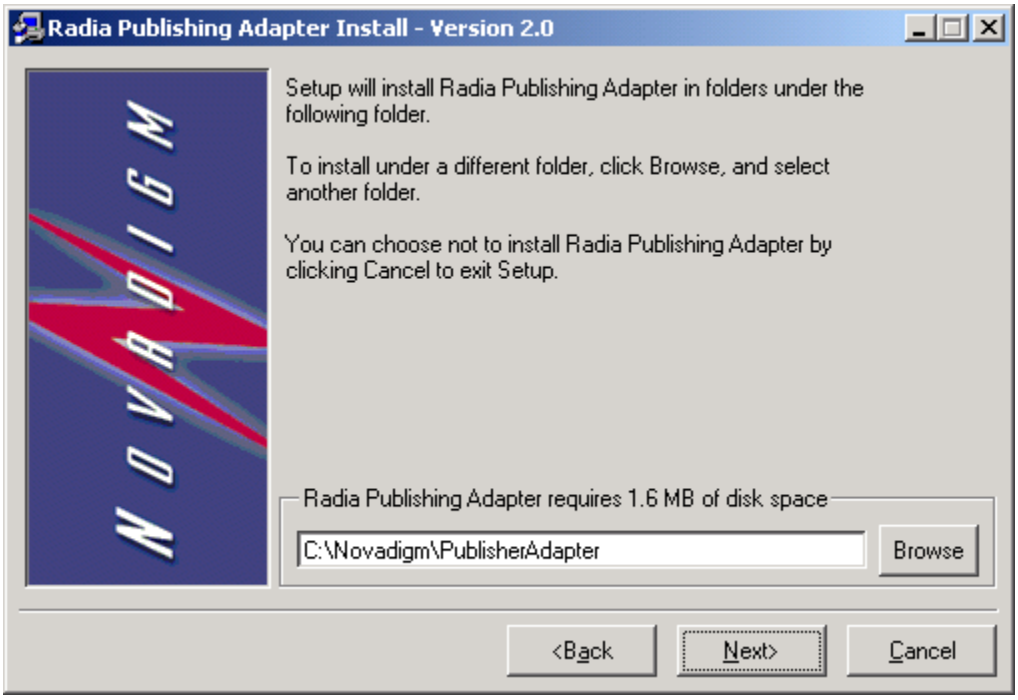

Figure 2.2 ~ Directory Location window of the Radia Publishing Adapter.

3. Type the name of the directory where you would like to install the Radia Publishing Adapter (default is `C:\Novadigm\PublisherAdapter`), or click **Browse** to select the location from the **Browse** dialog box.

4. Click **Next**.

   If the directory you specified already exists, the dialog box in Figure 2.3 on page 27 appears.

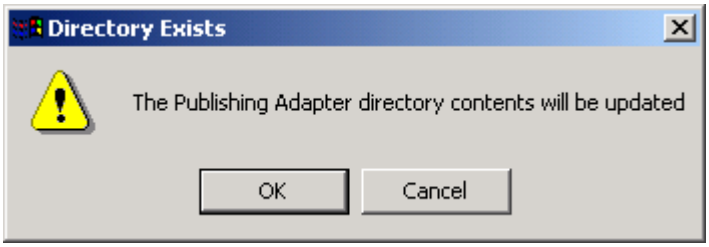

Figure 2.3 ~ Directory Exists message.

5. Click **OK**.

The **Installation Settings** window opens.

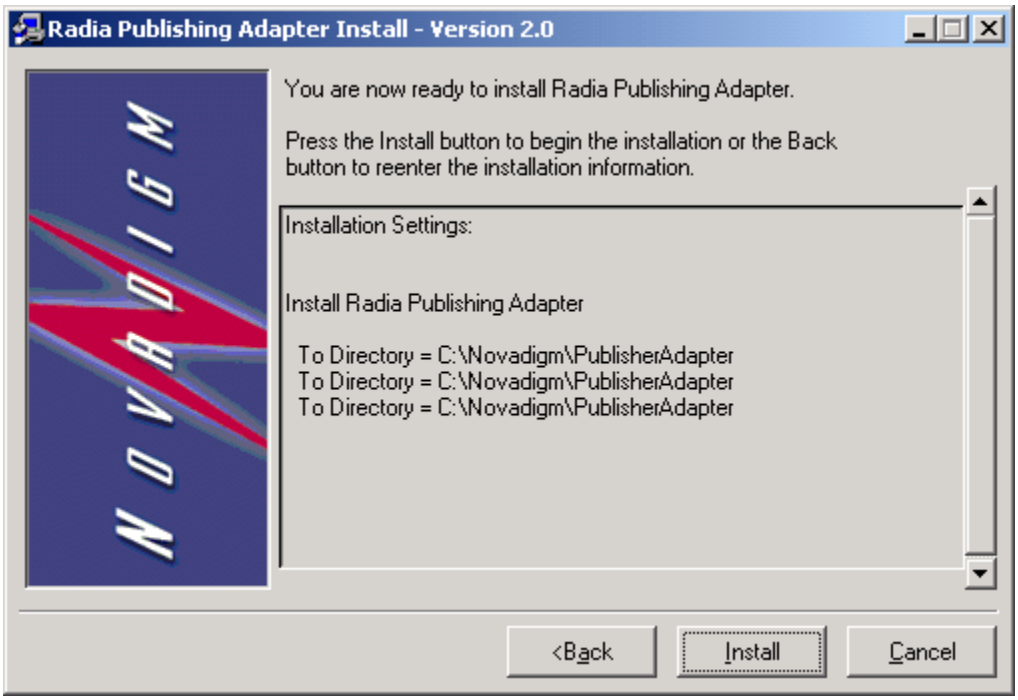

Figure 2.4 ~ Installation Settings window.

6. Click **Install**.

The **Installation Progress** window opens.

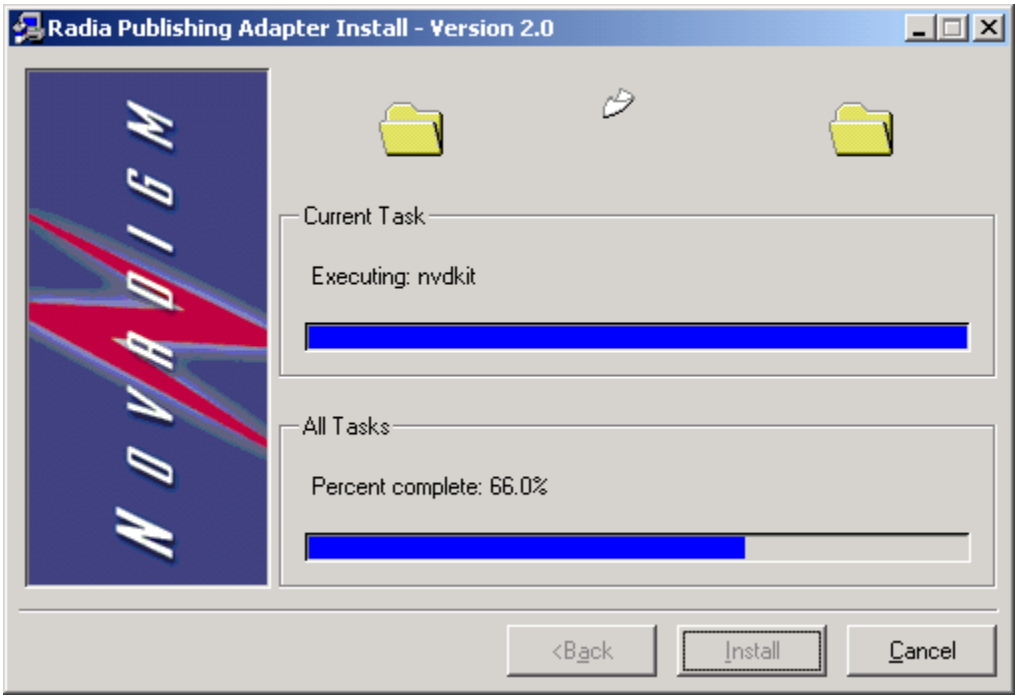

Figure 2.5 ~ Installation Progress window of the Radia Publishing Adapter.

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

You have successfully installed the Radia Publishing Adapter for Windows.

Installing the Radia Publishing Adapter for UNIX

If you are installing the Radia Publishing Adapter on a UNIX system that supports graphics, the graphical installation will automatically begin after it is started. For UNIX systems that support graphics, see *UNIX Graphical Installation* below. For UNIX systems that do not support graphics, the non-graphical installation program is automatically started. For UNIX systems that do not support graphics, see *UNIX Non-Graphical Installation* on page 34.

Note

If you are installing the Radia Publishing Adapter onto a UNIX system that supports graphics, but you would like to use the non-graphical mode instead, change your current directory to the location of the `install` program on the CD media and type:

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

This will start the non-graphical installation of the Radia Publishing Adapter. See *UNIX Non-Graphical Installation* on page 34 for instructions.

UNIX Graphical Installation

This section guides you through the graphical installation of the Radia Publishing Adapter.

To install the Radia Publishing Adapter 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.

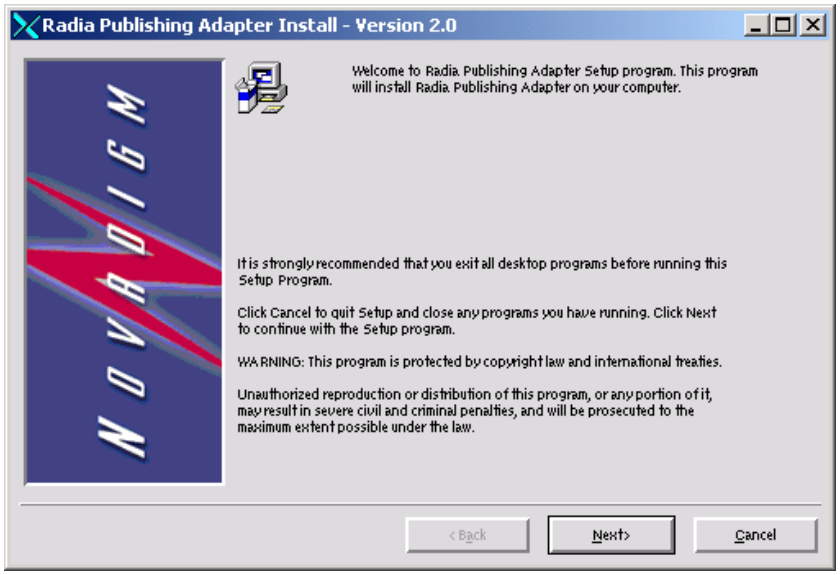

Figure 2.6 ~ Welcome window of the Radia Publishing Adapter.

3. Click **Next**.

The **Directory Location** window opens.

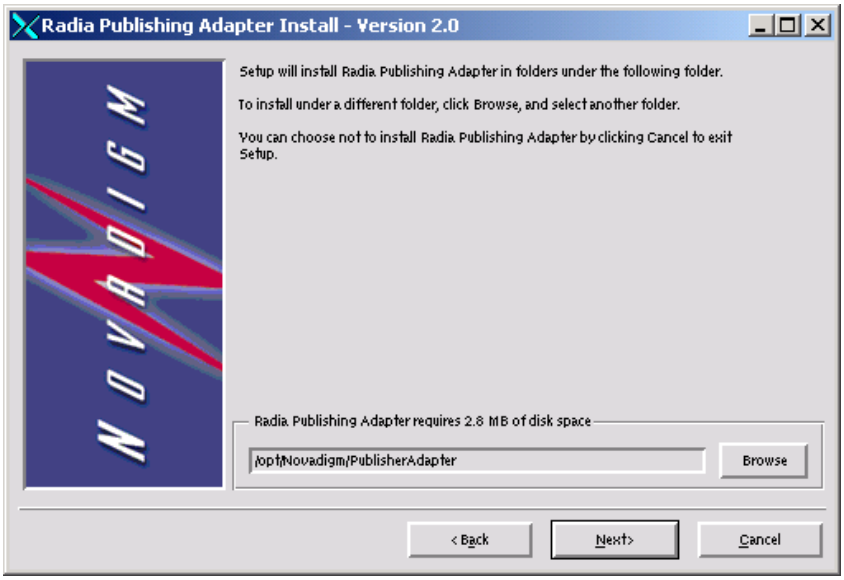

Figure 2.7 ~ Directory Location window of the Radia Publishing Adapter.

4. Type the name of the directory to which you would like to install the Radia Publishing Adapter (default is `/opt/Novadigm/PublisherAdapter`), or click **Browse** to select the location from the **Browse** dialog box.
5. Click **Next**.

   If the directory you specified already exists, the dialog box in Figure 2.8 below appears.

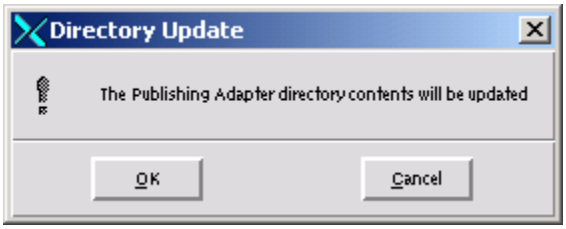

Figure 2.8 ~ Directory Update dialog box.

6. You can specify a new directory by clicking **Cancel** and returning to the previous step, or click **OK** to proceed.

The **Installation Settings** window opens.

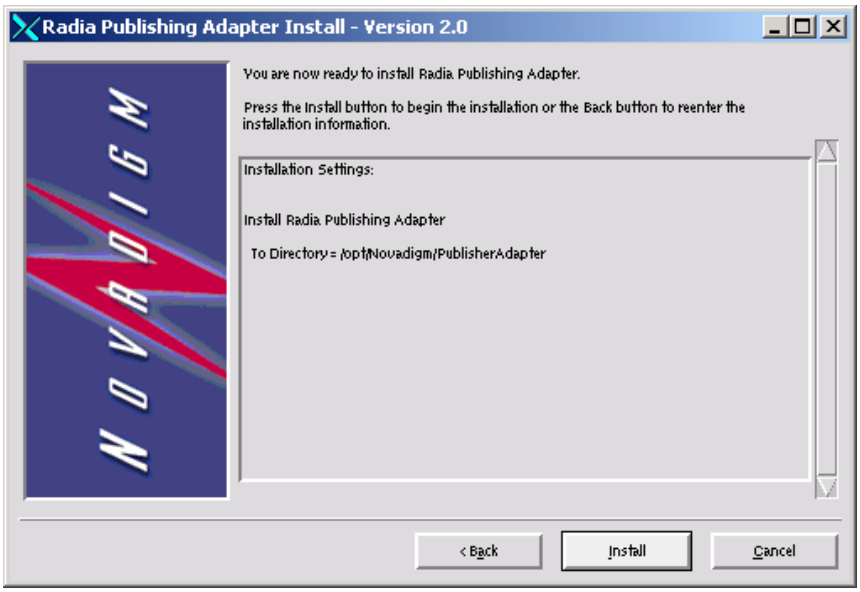

Figure 2.9 ~ Installation Settings window.

7. Click **Install**.

The **Installation Progress** window opens.

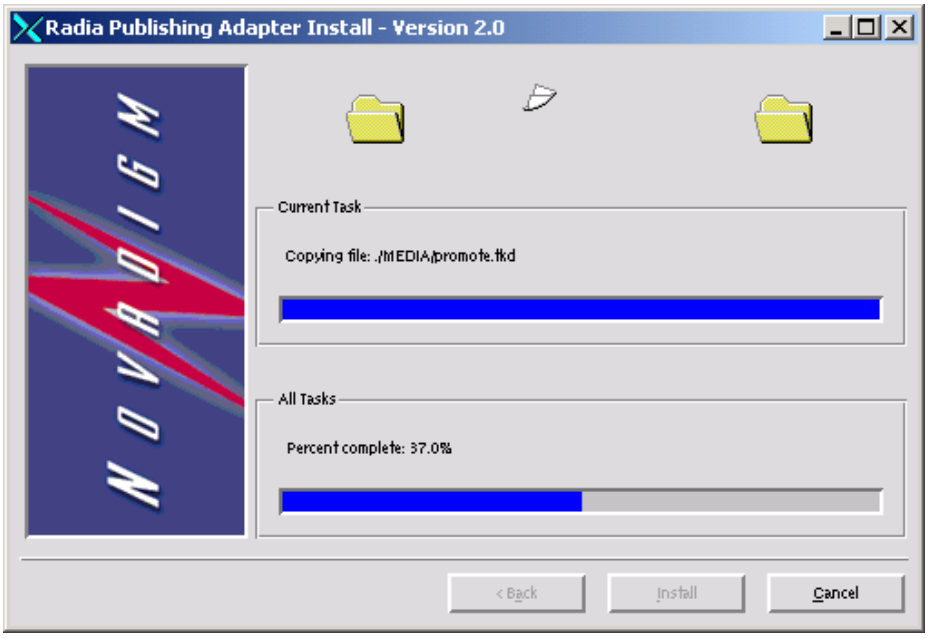

Figure 2.10 ~ Installation Progress window of the Radia Publishing Adapter.

8. When the installation is finished, click **Finish**.

You have successfully installed the Radia Publishing Adapter for UNIX.

UNIX Non-Graphical Installation

This section guides you through the non-graphical installation of the Radia Publishing Adapter for UNIX.

To install the Radia Publishing Adapter using the non-graphical installation

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

   The Radia Publishing Adapter installation begins.

```
Installing Radia Publishing Adapter
Welcome to Radia Publishing Adapter Setup program.
This program will install Radia Publishing Adapter on your computer.

It is strongly recommended that you exit all desktop programs
before running this Setup Program

Type Q to quit Setup and close any programs you have
running. Type C to continue with the Setup program.
(To exit install at any prompt, type <cancel>)

WARNING: This program is protected by copyright law and
international treaties.

Unauthorized reproduction or distribution of this program, or any
portion of it, may result in severe civil and criminal penalties,
and will be prosecuted to the maximum extent possible under the law.

Enter C to Continue with the installation or Q to Quit the setup program:
```

Figure 2.11 ~ Non-graphical installation of the Radia Publishing Adapter for UNIX.

3. Type **C**, and then press ENTER.

```
Setup will install Radia Publishing Adapter in folders under the following
folder.
To install under a different folder, enter another folder.
You can choose not to install Radia Publishing Adapter by exiting Setup.
Radia Publishing Adapter requires 2.8 MB of disk space
Default: /opt/Novadigm/PublisherAdapter
Directory name:
```

Figure 2.12 ~ Specify the location of the Radia Publishing Adapter

4. Accept the default location for the Radia Publishing Adapter (`/opt/Novadigm/PublisherAdapter`) by pressing ENTER, or specify a different location.

If the directory you specify already exists, you will be prompted to continue, as seen in Figure 2.13 below. If the directory does not exist, the installation program will display the Installation Settings, as seen in Figure 2.14 below.

```
The Publishing Adapter directory contents will be updated
Do you wish to continue? Y|N (Y):
```

Figure 2.13 ~ Existing directory will be updated.

5. Type **Y**, and then press ENTER.

```
You are now ready to install the Radia Publishing Adapter.
Installation Settings:

Install Radia Publishing Adapter

  To Directory = /opt/Novadigm/PublisherAdapter

Enter Y to begin the installation
Enter N to reenter the installation information.
Please enter your choice (Y):
```

Figure 2.14 ~ Installation Settings.

6. 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.

```
Starting Install . . .
Complete the configured install process? Y|N (Y):
```

Figure 2.15 ~ Complete the Radia Publishing Adapter installation.

7. To complete the configured installation process, press ENTER.

The Radia Publishing Adapter is now installed.

Summary

- The Radia Publishing Adapter is available for Windows and UNIX operating systems.
- Before installing the Radia Publishing Adapter, Novadigm recommends stopping any currently 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

One method available for publishing with the Radia Publishing Adapter is configuration file-based publishing. This method uses a configuration file (`promote.cfg`) that includes your publishing specifications. The publishing session is then executed from the command line. Command line parameters are described in Table 3.1 below and the configuration file is described in *The PROMOTE Configuration File* on page 41.

Execute the command line from the directory where you installed the Radia Publishing Adapter (default is `C:\Novadigm\PublisherAdpater\`). The command line is preceded with `nvdkit promote.tkd`, files that were installed during the Radia Publishing Adapter installation and contain the Novadigm runtime Tcl interpreter and configuration file-based publishing code respectively. Table 3.1 below shows the necessary command-line options.

EXAMPLE

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

Table 3.1 ~ Command-Line Parameters for promote.tkd

Parameter	Description
`-cfg` *filename*	Specifies the file that contains the configuration options for this execution of the Radia Publishing Adapter. 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*	Radia administrator user ID. The default is RAD_MAST. This parameter is optional.
`-pass` *password*	Radia administrator password. This parameter is optional.
`-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 3.2 below, describes the configuration file parameters. See the sample PROMOTE Configuration File in Table 3.3 on page 45.

The PROMOTE Configuration File Format

Table 3.2 ~ The 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.
service	Defines the name of the ZSERVICE class instance that will be optionally created (or updated) in the Radia Database 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 Radia Publishing Adapter 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 Radia Publishing Adapter whether to use compression. 1 = Use compression. 0 = Do not use compression.

Table 3.2 ~ The PROMOTE Configuration File Format (promote.cfg)

Option	Description
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.
insource	Specifies the name of the source file. The specified file should contain a list of qualified filenames, one per line, to be published. Additionally, numsplit and distroot can be specified in the file. These options behave in the same manner as described in the *filescan* section of this table. **Note**: Relevant only when intype=FILE. The formats that are accepted for the lines in the file are presented below: • **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 presented also.

Example A

```
global numsplit 3
global distroot d:/myapps
d:/temp/src/apps/a.dat
d:/temp/src/apps/test2.tcl

Output:

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

Example B

```
global numsplit 2
global distroot d:/place
d:/temp/list.pdf
d:/temp/mymk.tcl

Output:

  (distroot) |    (stem)
d:/place/list.pdf
d:/place/mymk.tcl
```

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 Radia Configuration Server. For example: radia://localhost:3464

Table 3.2 ~ The PROMOTE Configuration File Format (promote.cfg)

Option	Description
path	Defines the Radia Database 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. 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.

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

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. type - Identifies the type of Radia Configuration Server file being filtered. This value tells the publishing session how to create the instance in the Radia Database for a given file that matches the filtering criteria. Accepted values are FILE, DESKTOP, and REGISTRY.

Table 3.2 ~ The PROMOTE Configuration File Format (promote.cfg)

Option	Description
	`class` - Radia Database class to be used for files selected by `filters`. For example: FILE, DESKTOP, and REGISTRY. **Note**: Refer to the section, *Operating System Considerations* on page 20 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.
`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 Radia Publishing Adapter 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 47 for more information.

Sample PROMOTE Configuration File (promote.cfg)

Table 3.3 below presents a sample promote.cfg including the standard defaults.

Table 3.3 ~ Sample PROMOTE Configuration File (promote.cfg)

```
promote.cfg

# Radia Publishing Adapter 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        radia://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       {}
```

Table 3.3 ~ Sample PROMOTE Configuration File (promote.cfg)

```
        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
    # expression        expression 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   {
    }
```

Important Note: Remove this section if doing only one filescan function.

Specifying Additional Attributes

Use the Radia Publishing Adapter `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 Radia Publishing Adapter 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 Radia Publishing Adapter. 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. See Table 3.4 below for an edited configuration file (`promote.cfg`) displaying examples of additional attributes.

Table 3.4 below, displays an excerpt from a configuration file containing the `all` filter with an additional attribute section (`attr`):

Table 3.4 ~ Filter Including 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 created by the Radia Publishing Adapter do not have filters associated with them. To specify attributes for these class instances use the format shown in Table 3.6 on page 50, with the attributes and their values specified between the braces.

Table 3.5 ~ Additional Package, Service, and Path class instance attribute forma

```
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 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.

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

Table 3.6 below, displays an example of specifying connections for the ZSERVICE instance.

Table 3.6 ~ ZSERVICE connections specified

```
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 Radia Publishing Adapter 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`

Table 3.7 on page 51, displays an excerpt of the configuration file with the connection type attributes specified.

Table 3.7 ~ Connection Type Attributes Specified in a Configuration File

```
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 Radia Publishing Adapter promote.

Table 3.8 below displays an excerpt of the table presented in the log file.

Table 3.8 ~ Log file excerpt

```
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 Radia Publishing Adapter 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

Note

With this enhancement the use of a configuration file is no longer a required configuration argument.

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
```

For example:

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

Therefore an example of a Radia Publishing Adapter 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 doesn't 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
```

The filters argument must be used to specify the unique name of the filter. There can be multiple filters 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.

Table 3.9 ~ Sample Configuration File with Additional Attributes Specified

```
#
# Radia Automated Publishing Interface
#
# 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
    host       radia://localhost:3464
    path       PRIMARY.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        {}
```

Table 3.9 ~ Sample Configuration File with Additional Attributes Specified

```
        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     {/julie/test}
                attr   {
               ZCREATE     TESTSTART
                ZDELETE           TESTOVER
                }
    }
  expression {
    }
attr package {
```

Table 3.9 ~ Sample Configuration File with Additional Attributes Specified

```
            releASE    3.5.6
            wrong              thisiswrong
            includes   SOFTWARE.PACKAGE.ADAPT
            includes#2 SOFTWARE.PACKAGE.RAPILINK
          }
          attr  zservice {
          ZSVCMO          m
          URL             {WWW.NOVADIGM.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

Object-based publishing is accomplished through the use of `SCMAdapt.tkd`, a file included with your installation of the Radia Publishing Adapter. Using `SCMAdapt.tkd`, the Radia Publishing Adapter takes the input or output objects from one of the Novadigm legacy Source Control Management adapters (EDMPVCS or EDMATRIA), and publishes the specified files to Radia. This is done using command line arguments. Command line parameters are described in Table 4.1 below and the configuration file parameters are described in *The SCMADAPT Configuration File* on page 66.

Execute `SCMAdapt.tkd` on a command line from the directory where you installed the Radia Publishing Adapter (default is `C:\Novadigm\PublisherAdapter`). Once executed, `SCMADapt.tkd` uses the supplied arguments to determine the location of the objects and configuration file for the publishing session. Table 4.1 below describes the `SCMAdapt.tkd` command line parameters.

EXAMPLE

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

Table 4.1 ~ 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 Radia Publishing Adapter. 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 4.3 for a description of the configuration file parameters.
`-user` *userid*	Radia administrator user ID. The default is `RAD_MAST`. This parameter is optional.
`-pass` *password*	Radia administrator password. The default is " " (no password). This parameter is optional.
`-phase` *input*	This parameter is optional. 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.

Input Objects

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

ZPROMDFT Object	default values for the adapter.
and	
ZINPUT Object	input to the adapter. (Use the Radia Screen Painter or the Radia Client Explore to build input to `SCMAdapt`.)

OR

ZPROMDFT Object	default values for the adapter.
and	
ZPROMOTE Object	output of the adapter (output from a Novadigm legacy SCM Adapter).

Note

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 4.2 shows the variables of the ZPROMDFT object. Although all the variables listed might appear in a Novadigm legacy SCM adapter object, those marked N/A are not used by the Radia Publishing Adapter. Using an object editor (such as the Radia Client Explorer), open the ZPROMDFT object to ensure that the required variables are present.

Table 4.2 ~ 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 is suffixed with "*", 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
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 is to 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

Important Note

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 either 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. This variable is equivalent to ZLOCCLNT in EDM.

  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.

  Note

  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 detailed information on the SCMADAPT configuration file. The first section, *The SCMADAPT Configuration File*, presents a table in the format of the configuration file. The second section, *The SCMADAPT Configuration File Sample*, presents a sample `scmadapt.cfg`, showing the standard defaults.

The SCMADAPT Configuration File

Table 4.3 ~ The SCMADAPT Configuration File Parameters (scmadapt.cfg)

Option	Description
`compress`	Tells the Radia Publishing Adapter 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 Radia Configuration Server, for example, `radia://localhost:3464`.
`path`	Defines the Radia Database 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 (scmadapt.cfg)

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. The following table presents the commands of the Configuration File, as well as a description with guidelines for specifying.

A sample configuration file is given in Table 4.4 below.

Table 4.4 ~ Sample SCMADAPT Configuration File (scmadapt.cfg)

```
scmadapt.cfg

# Radia Publishing Adapter Default Options
#
# 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               radia://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

Radia Native Packaging

At the end of this chapter, you will:

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

What is Radia Native Packaging?

Radia Native Packaging is a feature of the Radia Publishing Adapter specifically designed for UNIX environments. Radia Native Packaging is a command-line driven content-publishing tool supporting native HP-UX, Solaris, AIX, and RedHat Linux software; it is not a graphical publishing tool nor a mainstream publisher tool. Radia Native Packaging is installed during the regular installation of the Radia Publishing Adapter on a UNIX system.

Radia Native Packaging explores UNIX native software depots, searches for available native packages and publishes wrapped native packages to the Radia Configuration Server. Radia Native Packaging will publish all necessary information that will allow you immediate installation of native software to end clients.

Additionally, Radia Native Packaging publishes information about native package dependencies and will optionally include them with a published package.

Why use Radia Native Packaging?

Radia Native Packaging supports HP-UX (SD), Solaris (SVR4 and patches), AIX (bff), and RedHat Linux RPM software package formats. With the use of Radia 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 clients.

This document assumes that the system administrator using the Radia Native Packager possesses packaging/publishing knowledge for a Radia infrastructure database.

Overview

Radia Native Packaging creates the standard instances of ZSERVICE, PACKAGE, and PATH in the SOFTWARE domain of the Radia Database. Radia 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). Two other classes (SDDEP and SVR4DEP) hold instances that describe native package dependency relations.

For each native software package selected, Radia 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. Additionally it will create an instance of the PACKAGE class that will contain the newly created SD/SVR4/SOLPATCH/AIX/RPM instance and an instance of ZSERVICE class that contains the new PACKAGE instance. In case the selected native software package contains any dependency relationships, that information is published into an instance of SDDEP or SVR4DEP class.

Important Note

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

Radia Native Packaging System Requirements

Radia 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 Radia Native Packaging.
- Network connectivity to the Radia Configuration Server.
- Space on `/tmp` file system for temporary depot files used for publishing.

Note

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

Required Classes

Radia Native Packaging requires specific classes for each operating system. Make sure your Radia Database includes these SOFTWARE domain classes before using Radia Native Packaging.

Table 5.1 ~ Prerequisite Classes

Operating System	Class
Solaris	SVR4 Packages (SVR4) Solaris Patches (SOLPATCH)
HP-UX	SD Packages (SD)
AIX	IBM AIX Packages (AIX)
RedHat Linux	Linux RPM Packages (RPM)

Radia Native Packaging and the Radia Client

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

Note

Radia Client version 3.0 or higher is required to deploy packages published to the database using Radia Native Packaging. Contact your Novadigm sales representative for more information.

Radia Native Packaging Command-Line Interface

Radia Native Packaging is run from the command line. The base input parameter for Radia Native Packaging is the source depot containing HP-UX, Solaris, AIX, or RedHat Linux software. Note that the depot needs to be in a normal format (disk depot). In addition, you can specify the selection of the software you want to publish, and in the event Radia Database user verification is enabled, an optional user ID and password can be designated.

```
Usage: rnp –d depot_path –m manager_ip:manager_port
               [-user user_id] [-pass password] [-l logfile] [-help]
               [-i] [-a | -p package1[,r=revision][,a=arch][,v=vendor]
                           -p package2[,r=revision][,a=arch][,v=vendor] …]

                            [-P] [-r] [-f prefix]
                             [-s] [-t svc_type]
```

Figure 5.1 ~ Command-Line usage for Radia Native Packaging.

Table 5.2 ~ Command-Line Parameters for Radia Native Packaging on page 73 contains the description of the command line arguments for Radia Native Packaging.

Table 5.2 ~ Command-Line Parameters for Radia Native Packaging

Parameter	Description
`-d` *depot path*	Specifies the path to the depot containing SVR4 packages, SD native software, Solaris Patches, AIX bff or RedHat RPM Packages. Software contained in this depot will serve as an input to Radia Native Packaging for wrapping native software. This parameter is required.
`-m` *ipaddr:port*	Specifies the IP address and port of the Radia Configuration Server you intend to publish software to. This parameter is required.
`-a`	Specifies to publish all native software available in the depot. This parameter is optional. You cannot use this parameter together with `-p`.
`-p` *package* [`,r`=revision] [`,a`=arch] [`,v`=vendor]	Specifies a software package to publish to the Radia Configuration Server. Specify one of 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, • or specify 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
-i	Instructs Radia Native Packaging to include prerequisite software package (supported for SD and SVR4 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. This is not supported when promoting Solaris Patches, Linux RPM packages, or AIX bff software packages.
`-P`	Instructs Radia Native Packaging to publish Solaris Patches instead of SVR4 packages on SunOS. This parameter is optional and available only for Solaris.
`-r`	Instructs Radia Native Packaging to publish a response file for Solaris non-interactive installation support. A <pkgname>.response file which must reside in `depot_path` is promoted together with the Solaris package file. If the response file doesn't exist the package is promoted without it. Available only on Solaris systems.
`-user` *user ID*	Radia administrator user ID. The default is RAD_MAST. This parameter is optional.
`-pass` *password*	Radia administrator password. This parameter is optional.
`-l` *logfile*	Instructs Radia Native Packaging to store the log in the logfile specified. If this option is omitted, the default log file created is `promote.log`. This parameter is optional.
`-f` *prefix*	Instructs Radia 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.

Table 5.2 ~ Command-Line Parameters for Radia Native Packaging

Parameter	Description
`-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.
`-s`	Instructs Radia Native Packaging to skip the creation of services for the packages to be published.
`-help`	Display help on the command-line usage and the `rnp.cfg` configuration file format.

Radia Native Packaging Options File (`rnp.cfg`)

If you usually use the same source depot or publish to the same Radia Configuration Server you can create a file, `rnp.cfg`, in the same directory where you have the Radia Native Packaging components installed. Add default option values in the following format:

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

Note

By default, `rnp.cfg` is not supplied.

Table 5.3 ~ 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 Configuration Server	None
`manager_port`	Port number of the Radia Configuration Server	`manager_port=3464`
`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`

Table 5.3 ~ Supported `rnp.cfg` Settings and Default Values

Setting	Expected Values	Default Value
`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 a optional service.	`service_type=M`
`select_patches`	`select_patches=[yes/no]` A value of `yes` shall 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`
`include_dependencies`	`include_dependencies=[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_dependencies=no`
`user`	`user=userid` Administrator ID used for authentication with the Radia Configuration Server.	`User=RAD_MAST`
`password`	`password=pass` Administrator password, used for authentication with the Radia Configuration Server.	blank

Publishing with Radia Native Packaging

Examples

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

1. Change your current working directory to the Radia Publishing Adapter directory (default `/opt/Novadigm/PublisherAdapter/`).
2. On the command line, type:

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

   Refer to *Table 5.2 ~ Command-Line Parameters for Radia Native Packaging* on page 73 for an explanation of the Radia Native Packager command-line parameters.

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

1. Change your current working directory to the Radia Publishing Adapter directory (default `/opt/ Novadigm/PublisherAdapter/`).
2. On the command line, type:

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

   Refer to *Table 5.2 ~ Command-Line Parameters for Radia Native Packaging* on page 73 for an explanation of the Radia Native Packager command-line parameters.

To publish a specific Solaris Patch residing in the specified depot

1. Change your current working directory to the Radia Publishing Adapter directory (default `/opt/ Novadigm/PublisherAdapter/`).
2. On the command line, type:

   ```
   ./rnp -d /var/spool/patch -p 111111-03 -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 Radia Publishing Adapter directory (default `/opt/ Novadigm/PublisherAdapter/`).
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 Radia Publishing Adapter directory (default `/opt/ Novadigm/PublisherAdapter/`).
2. On the command line, type:

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

Wrapped Native Packages

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

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

The supplied client methods are designed to invoke the native software management utilities, therefore, the methods are not interchangeable between client platforms. For example: The file `sd.tcl` supplied with HP-UX Radia clients 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.

The depot containing native software in compressed format is promoted to SD/SVR4/SOLPATCH/AIX or RPM class (class is similar to UNIXFILE class). Table 5.4 below lists the modified attributes:

Table 5.4 ~ SD/SVR4/SOLPATCH/AIX/RPM Class Instance Attributes Modified by Radia 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 complete software spec on HP-UX (tag, version, architecture and vendor) and SVR4 package name on Solaris, filename without `bff` suffix on AIX (all dots are converted to underscores), the RPM Package Name on RedHat Linux.
ZRSCCFIL	Specifies path to the file that is included in this instance. The file contains the native packaged software.
AUTOBOOT	This Boolean variable is set to `Y` in the SD class instance in case you have wrapped SD software that contains reboot file set.

Table 5.4 ~ SD/SVR4/SOLPATCH/AIX/RPM Class Instance Attributes Modified by Radia Native Packaging

Attribute	Description
ZCREATE	Uses method "Harness" call. The client side script contains a native command call to install software: • HP-UX: `swinstall -s &LOCATION/&ZRSCCFIL &ZRSCNAME` • Solaris Package: `pkgadd -d &LOCATION/&ZRSCCFIL -n &ZRSCNAME` Patch: `patchadd &LOCATION/&ZRSCNAME` • AIX: `installp -a -d &LOCATION &ZRSCFLEV` • Linux: `rpm -i &LOCATION &ZRSCCFIL` **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 client side script contains a native command call to remove software: • HP-UX: `swremove &ZRSCNAME` • Solaris Package: `pkgrm -n &ZRSCNAME` Patch: `patchrm &ZRSCNAME` • AIX: `installp -ug &ZRSCFSET` • Linux: `rpm -e &ZRSCNAME` **Note:** On HP-UX, when a native software application is removed, the application files are deleted, but the directory structure will remain.
ZUPDATE	Uses method "Harness" call. The client side contains a native command call to update software: • HP-UX: `swinstall -x reinstall=true -s /&ZRSCCFIL &ZRSCNAME` • Solaris Package: `pkgadd -d /&ZRSCCFIL -n &ZRSCNAME` Patch: `showrev -p; patchadd &LOCATION/&ZRSCNAME` • AIX: `installp -a -d &LOCATION &ZRSCFLEV` • Linux: `rpm -i -replacefiles -replacepkgs &LOCATION/&ZRSCNAME`
ZVERIFY	Uses method "Harness" call. The client side script contains a native command call to verify installed software: • HP-UX: `swverify &ZRSCNAME` • Solaris Package: `pkgchk -n &ZRSCNAME` Patch: `showrev -p` • AIX: `lppchk -v &ZRSCFSET` • Linux: `rpm -V &ZRSCNAME`

Table 5.4 ~ SD/SVR4/SOLPATCH/AIX/RPM Class Instance Attributes Modified by Radia Native Packaging

Attribute	Description
ZREPAIR	Uses method "Harness" call. The client side script contains a native command call to repair installed software (reinstall): • HP-UX: `swinstall -x reinstall=true -s /&ZRSCCFIL &ZRSCNAME` • Solaris Package: `pkgadd -d /&ZRSCCFIL -n &ZRSCNAME` Patch: `patchadd &LOCATION/&ZRSCNAME` • AIX: `installp -a -d &LOCATION &ZRSCFLEV` • Linux: `rpm -i -replacefiles -replacepkgs &LOCATION/&ZRSCNAME`
ADDDEPS	HP-UX only. Auto-select dependencies set to N by default.
RESPONSE (Solaris SVR4 package only)	Specifies if response file for this package exists (`Y/N`). Default value is `N`.
RESPOBJ (Solaris SVR4 package only)	Specifies whether this is the response file (Y/N). Default value is N.
RESPONSE (Solaris SVR4 package only)	Path to response file (if it exists). Default is empty string.
COMMIT (AIX only)	If turned on, this flag commits all file sets in package update. Default value is `N`.
FORCE (AIX only)	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 (AIX only)	Name of prerequisite AIX file sets. Informational attribute only.
COREQ (AIX only)	Name of co requisite AIX file sets. Informational attribute only.
INSTREQ (AIX only)	Name of installed requisite AIX file set. Informational attribute only.
IFREQ (AIX only)	Name of if requisite AIX file set. Informational attribute only.
PKGVER (RPM only)	RPM Package Version. Informational attribute only.
PKGREL (RPM only)	RPM Package Release. Informational attribute only.
PKGARCH (RPM Only)	RPM Package Architecture. Informational attribute only.
PKGSUMM (RPM Only)	RPM Package Summary. Informational attribute only.

Table 5.4 ~ SD/SVR4/SOLPATCH/AIX/RPM Class Instance Attributes Modified by Radia Native Packaging	
Attribute	**Description**
REQPKGS (RPM Only)	RPM Required Packages. Informational attribute only.
REQCMDS (RPM Only)	RPM Package Required Commands. Informational attribute only.
REQLIBS (RPM Only)	RPM Package Required shared libraries. Informational attribute only.

An instance of PACKAGE class is created that contains the instance of SD/SVR4/SOLPATCH/AIX/RPM class. Table 5.5 below describes how Radia Native Packaging maps native package information into Radia PACKAGE class attributes.

Table 5.5~ PACKAGE Class Attributes Modified by Radia Native Packaging	
Attribute	**Description**
Instance Name	On HP-UX Radia 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 AIX, 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`).
RELEASE	SD revision, SVR4 and RPM version native attributes are mapped into RELEASE. On AIX no release information is mapped since it is usually included in package name (filename). For Solaris Patches, the patch revision is mapped into RELEASE.
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 or SOLPATCH_5_8_<PATCH>`). On AIX, `AIX_` prefix is added to AIX package name without the `bff` suffix. On RedHat Linux, `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>`).

Table 5.5~ PACKAGE Class Attributes Modified by Radia Native Packaging

Attribute	Description
DESCRIPT	SD's title and SVR4's name attributes are mapped into DESCRIPT. For SOLPATCH, the Patch 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.
ZSTOP001	On HP-UX possibly contains SD products target OS release.
FILE	Holds reference to respective instance of SD/SVR4/SOLPATCH/AIX/RPM class.

Radia Native Packaging also creates an instance of ZSERVICE class holding previously created instance of PACKAGE class. *Table 5.6 ~ ZSERVICE Class Attributes Modified by Radia Native Packaging* below lists the modified attributes.

Table 5.6 ~ ZSERVICE Class Attributes Modified by Radia Native Packaging

Attribute	Description
Instance Name	Instance name is composed differently on HP-UX and Solaris. On HP-UX Radia Native Packaging will take SD product tag, pre-pend SD_ to it and append a date and sequence number to guarantee uniqueness (SD_<tag>_yyyymmddn). On Solaris only the SVR4_ string is pre-pended to the SVR4 package name (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 AIX, 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).
VERSION	SD revision or SVR4 version native attributes are mapped into VERSION. For Solaris Patches, the patch revision is mapped into VERSION. On AIX no release information is mapped since it is usually included in package name (filename). On RedHat Linux, the RPM Package Version is mapped into VERSION.

Table 5.6 ~ ZSERVICE Class Attributes Modified by Radia Native Packaging

Attribute	Description
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, the name is composed of `SOLPATCH_` followed by Patch Synopsis. On AIX, the `AIX_` prefix is added to the AIX package name without `bff` suffix. On RedHat Linux, 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` suffix. On RedHat Linux, 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 SD product or SVR4 package has any dependency relations to other software, this information is captured in the SDDEP or SVR4DEP classes. *Table 5.6 ~ ZSERVICE Class Attributes Modified by Radia Native Packaging* on page 81 and *Table 5.8 ~ SVR4DEP Class Attributes Modified by Radia Native Packaging* on page 83 describe the modified attributes.

Table 5.7 ~ SDDEP Class Attributes Modified by Radia Native Packaging

Attribute	Description
PKGNAME	Holds software spec of SD product.
PREREQ	Software spec of prerequisite SD product. SD's dependencies are on the fileset level. Since Radia Native Packaging wraps SD products dependencies are elevated to product level.
COREQ	Software spec of corequisite SD product.
EXREQ	Software spec of exrequisite SD product.

Table 5.8 ~ SVR4DEP Class Attributes Modified by Radia Native Packaging

Attribute	Description
PKGNAME	Holds the name of SVR4 package.
PREREQ	Name of prerequisite SVR4 package.
INCOMP	Name of incompatible SVR4 package.
REVERSE	Name of SVR4 package with reverse dependency to base package.

Note

If a package requires a system reboot after a Client Connect, make sure the `handle_reboot radskman` parameter is set to `Y`. See the *Radia Application Manager Guide for UNIX,* for more information.

Automatic Inclusion of Prerequisite Packages

If you specify the `-i` command line option, Radia Native Packaging will include prerequisite packages into the depot with the package you are publishing to Radia. The prerequisite package needs to exist in the depot Radia Native Packaging is using as a source. This feature is not supported for AIX (bff) packages, Solaris Patches, or RedHat Linux (rpm) packages.

The advantage of using this option is that the installation of native software packages will not fail because of a missing prerequisite package. On the other hand if a prerequisite package is already installed you are using more network bandwidth and disk space than necessary.

Note

This feature is not supported for AIX (bff) packages, Solaris Patches, or RedHat Linux (rpm) packages.

Summary

- Radia Native Packaging is a feature of the Radia Publishing Adapter specifically designed for UNIX environments.
- Radia Native Packaging requires specific classes for each operating system.

Lists

Figures

Tables

Procedures

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Z