



ZENworks[®]

Patch Management

User Guide

Novell Patch Developers Kit v6.4

Novell[®]

02_007N_6.4.2.10

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
Phone: 800.858.4000
www.novell.com

Copyright © 1997-2007 PatchLink® Corporation. ALL RIGHTS RESERVED. U.S. Patent No. 6,990,660, Other Patents Pending. This manual, as well as the software described in it, is furnished under license. No part of this manual may be reproduced, stored in a retrieval system, or transmitted in any form—electronic, mechanical, recording, or otherwise—except as permitted by such license.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: PATCHLINK® CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES IN REGARDS TO THE ACCURACY OR COMPLETENESS OF THE INFORMATION PROVIDED IN THIS MANUAL. PATCHLINK® CORPORATION RESERVES THE RIGHT TO MAKE CHANGES TO THE INFORMATION DESCRIBED IN THIS MANUAL AT ANY TIME WITHOUT NOTICE AND WITHOUT OBLIGATION TO NOTIFY ANY PERSON OF SUCH CHANGES. THE INFORMATION PROVIDED IN THE MANUAL IS NOT GUARANTEED OR WARRANTED TO PRODUCE ANY PARTICULAR RESULT, AND THE ADVICE AND STRATEGIES CONTAINED MAY NOT BE SUITABLE FOR EVERY ORGANIZATION. NO WARRANTY MAY BE CREATED OR EXTENDED WITH RESPECT TO THIS MANUAL BY SALES REPRESENTATIVES OR WRITTEN SALES MATERIALS. PATCHLINK® CORPORATION SHALL NOT BE LIABLE FOR ANY LOSS OF PROFIT OR ANY OTHER DAMAGES ARISING FROM THE USE OF THIS MANUAL, INCLUDING BUT NOT LIMITED TO SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR OTHER DAMAGES

Trademarks:

PatchLink™, PatchLink.com™, securing the enterprise™, WebConsole™, PatchLink Update™, PatchLink Quarantine™, PatchLink Enterprise Reporting Services™, PatchLink Scanner Integration Module™, PatchLink Developers Kit™, and their associated logos are registered trademarks or trademarks of PatchLink® Corporation.

Novell, Novell ZENworks®, Novell ZENworks® Patch Management Server, and Novell Agent are registered trademarks or trademarks of Novell, Inc.

RSA Secured® is a registered trademark of RSA Security Inc.

Apache is a trademark of the Apache Software Foundation

In addition, other companies' names and products mentioned in this document, if any, may be either registered trademarks or trademarks of their respective owners.

Feedback:

Your feedback lets us know if we are meeting your documentation needs. E-mail the Novell Technical Publications department at techpubs@patchlink.com to tell us what you like best, what you like least, and to report any inaccuracies.



Table of Contents

Table of Contents	iii
Preface	vii
About This Guide	vii
Document Conventions	viii
Chapter 1: Getting Started	1
Patch Developers Kit Overview	1
Defining Patch Structure	2
Vulnerabilities	2
Signatures	2
Fingerprints	2
Pre-requisites	3
Packages	3
Installing the PDK Components	3
Installing the PDK Application	6
Connecting ZENworks Patch Management Server and the PDK	9
Accessing the PDK	9
Chapter 2: Defining the Properties	11
Building the Vulnerability Process	11
Viewing the Vulnerability Properties Window	12
Vulnerability Properties Window Description	13
Creating and Editing Vulnerabilities	15
Working with Vendors	16
Chapter 3: Working with Signatures	19
Defining Signatures and Pre-Requirement Signatures	19
Defining the Signature and Pre-Requirement Signature Properties	20
Working With Signatures	21
Adding a Signature	21
Editing a Signature	23
Removing a Signature	23



Chapter 4: Creating Fingerprints _____ **25**

Viewing the Fingerprint Summary Window	25
Fingerprint Types	26
Adding Fingerprints	27
Adding Multiple Fingerprints	29
Removing a Fingerprint	30
Creating Fingerprints in Expert Mode (XML)	30
Using Fingerprint Types	31
Using the File Fingerprint	32
Fingerprint File XML Example	34
LogicalNOT XML Example Script	35
Using the Registry Fingerprint	35
Registry XML Example	36
LogicalNOT XML Example Script	36
Using the WMI Fingerprint	37
WMI XML Example	38
Using the SystemInfo Fingerprint	38
SystemInfo XML Example	39
Using the Expression Fingerprint	39
Expression XML Example	40
Using the Patch Fingerprint	41
Patch XML Example	42
Using the Script Fingerprint	42
Script XML Example	44

Chapter 5: Working With Packages _____ **45**

Defining Packages	45
Viewing the Package Properties	47
Defining Package Content	49
Working with Package Scripts	49
Adding a New Package	51
Adding an Existing Package	53
Removing a Package	53
Editing a Package	54
Adding Files to a Package	55
Adding a New Drive to a Package	56
Adding a New Macro to a Package	57
Creating a Folder for a Package	59
Inserting a Folder into a Package	59
Inserting Files into a Package	60
Deleting a File from a Package	62
Renaming a File within a Package	62
File Properties for a Package	63



Adding Package Scripts	64
Viewing Scripts With Notepad	65
Setting Package Deployment Flags	65
Package Flag Descriptions	65

Chapter 6: Working with Pre-Requisites _____ 67

Adding a Pre-Requisite	67
Adding an Existing Pre-Requisite	69
Removing a Pre-Requisite	70
Editing a Pre-Requisite	70

Chapter 7: Importing and Exporting Patches _____ 71

Defining the Import Summary Window	71
Import Summary Toolbar	72
Defining the Menu Options	72
Defining the Import Summary fields	73
Importing Patches	75
Using Command Line Importing	78
Exporting Patches	79

Appendix A: Reference: PLCCAgent Object Methods _____ 85

Defining the PLCCAgent Object Methods	85
GetOSVersion	85
GetPolicy	86
InitiateSystemShutdown	87
PollHost	88
RegCloseKey	89
RegEnumKey	90
RegEnumValue	91
RegOpenKey	93
RegQueryValue	94
RegRead	96
RegSetValue	98
SetReturnCode	100
Write	101

Appendix B: Index _____ 103





Preface

This ZENworks® Patch Management User Guide is a resource written for all users of ZENworks Patch Management. This guide defines the concepts and procedures for installing and implementing a successful installation of ZENworks Patch Management.

About This Guide

This guide contains the following chapters:

- Chapter 1, “Getting Started”
- Chapter 2, “Defining the Properties”
- Chapter 3, “Working with Signatures”
- Chapter 4, “Creating Fingerprints”
- Chapter 5, “Working With Packages”
- Chapter 6, “Working with Pre-Requisites”
- Chapter 7, “Importing and Exporting Patches”
- Appendix A, “Reference: PLCCAgent Object Methods”



Tip: This document is updated on a regular basis. To acquire the latest version of this document please refer to the Novell Support Web site (www.novell.com/support)



Document Conventions




The following conventions are used throughout this document to help you identify various information types:

Table 1.1 Document Conventions

Convention	Usage
bold	Command names, database names, options, wizard names, window and screen objects (i.e. Click the OK button)
<i>italics</i>	New terms, variables, and window and page names
UPPERCASE	SQL commands and keyboard keys
monospace	File names, path names, programs, executables, command syntax, and property names

The icons used throughout this document identify the following types of information:

Table 1.2 Icons Used

Icon	Alert Label	Description
	Note:	Identifies paragraphs that contain notes or recommendations.
	Tip:	Identifies paragraphs that contain tips, shortcuts, or other helpful product information.
	Warning:	Identifies paragraphs that contain vital instructions, cautions or critical information.



1 Getting Started

The **Patch Developers Kit (PDK)** allows administrators the ability to build custom patches to maintain software within their organization

Novell Patch Management Server provides a comprehensive solution to patching and maintaining a company network, and comes with a subscription of pre-built patches delivered over a secure Internet connection.

In this Chapter

- “Patch Developers Kit Overview” on page 1
- “Defining Patch Structure” on page 2
- “Installing the PDK Components” on page 3
- “Installing the PDK Application” on page 6
- “Connecting the Server and the PDK” on page 10
- “Accessing the PDK” on page 9

Patch Developers Kit Overview

The Novell PDK performs the following functions:

- Import patches
- Export patches



Defining Patch Structure

The structure of a Vulnerability allows the ability to create one patch applicable for many different operating systems and software versions. This allows for different packages and signatures capable of identifying the presence of patch files within a device.

As depicted in the following diagram, for each vulnerability you can have more than one signature. For each signature, you can have multiple fingerprints and pre-requisites. However, you can only have one package assigned per signature.

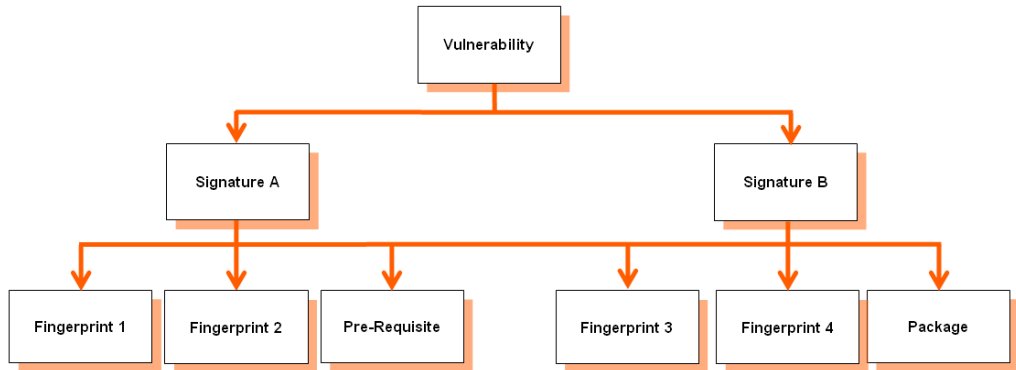


Figure 1.1 Patch Structure

Vulnerabilities

A vulnerability is the container for the entire object. All properties set for the vulnerability are viewed in the *Vulnerabilities* page of ZENworks Patch Management Server. Each vulnerability can have one or more signatures.

Signatures

Signatures recognize specific combinations of installed software in an operating system. Vulnerabilities usually contain multiple signatures to compensate for variances within applications. Frequently, a patch will require different executables, dynamic-link libraries, and switches in order to run or detect the patch within different operating systems.

Fingerprints

A fingerprint can represent a unique file, folder, registry key, or other data value somewhere within a system. Each signature can contain one or more fingerprints detecting if a patch is present in the system.



Pre-requisites

A pre-requisite is a signature belonging to another vulnerability with its own fingerprints. Adding a pre-requisite to a signature requires the pre-requisite be met before analyzing the signature for the current patch. If that signature's pre-requisite is met, the agent will analyze the fingerprints of the current signature, otherwise they will be ignored and the patch will not be applied to the device.

Packages

The package contains the actual files used to update or install software on the system. Each package contains the script commands for installing the package files or running the executable that installs the patch.

Installing the PDK Components

Before installing the PDK, you need the following items:

- Patch Management Server installed on the targeted server
- PDK Database installed



Note: The PDK database must be installed in the same location as Patch Management Server.

Installing the PDK database

1. Download the `Novell Patch Developers Kit 6.3 Database.msi` to the same server location ZENworks Patch Management Server is installed.



- From the downloaded location, select Novell Patch Developers Kit 6.3 Database.msi to extract the PDK database installshield wizard. The *Database Install Welcome* page opens.

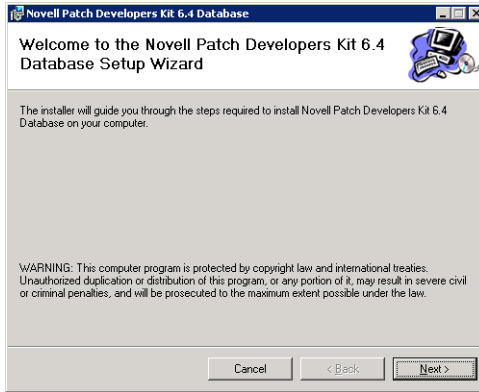


Figure 1.2 PDK Database Welcome page

- Click **Next**. The *License Agreement* page opens.



Figure 1.3 License Agreement page



4. Review the EULA. If you agree with the terms, select **I Agree** and click **Next**
The *Installation confirmation* page opens

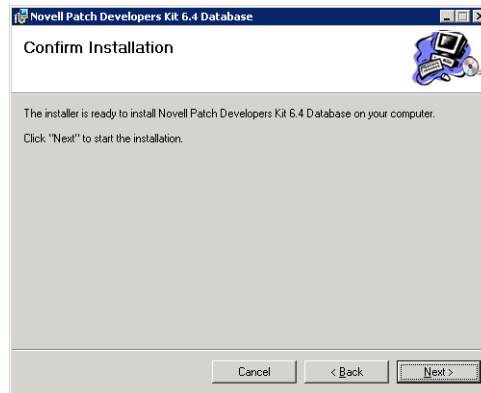


Figure 1.4 Installation Confirmation page

5. Click **Next**
The database installs and the *Installation Complete* page opens.

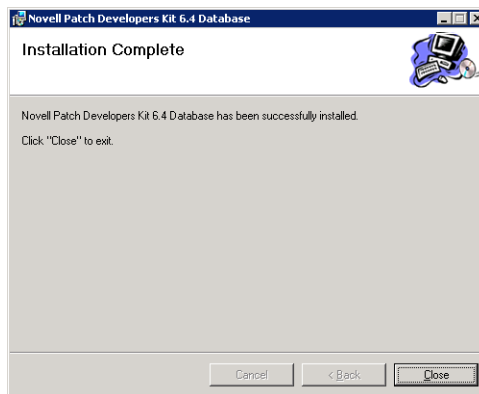


Figure 1.5 Installation Complete page

6. Click **Close** to complete the installation.



Installing the PDK Application

The PDK can be installed on any computer that has access to ZENworks Patch Management Server. The PDK must be installed on a computer configured with English (United States) as the language parameter.



Note: Any PDK version prior to 1.4 must be uninstalled before installing PDK 6.4.

Installing the PDK

1. Verify that you can connect to ZENworks Patch Management Server, and that the PDK database is installed in the same location.



Note: The PDK database file should be installed on the same location as ZENworks Patch Management Server. Go to [“Installing the PDK database”](#) for instructions on how to install the database.

2. From the downloaded location, select Novell Patch Developers Kit 6.3.msi to extract the Patch Developers Kit *InstallShield Wizard*. The *Agent Install Welcome* page opens.



Figure 1.6 PDK Install Welcome page



3. Click **Next** to proceed to the *License Agreement* page

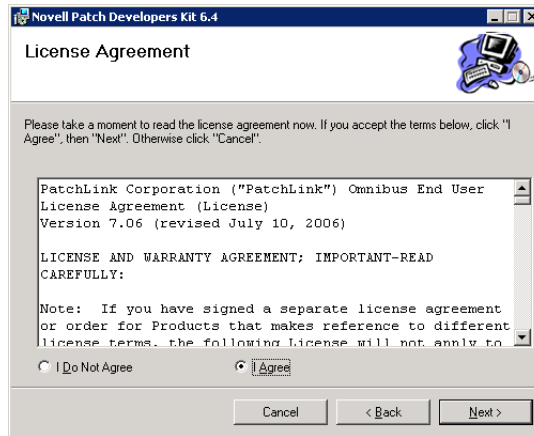


Figure 1.7 License Agreement Page

4. Review the EULA. If you agree with the terms, select **I Agree**, and click **Next**. The *Select Installation Folder* page opens.

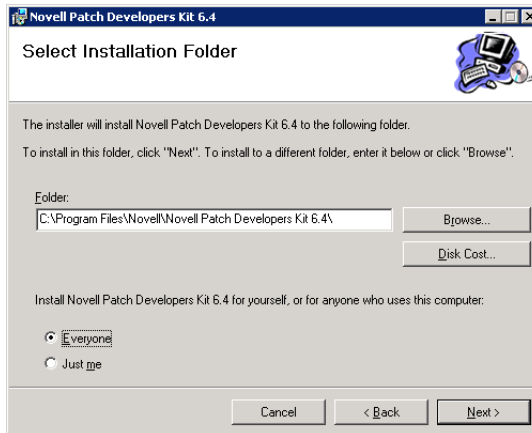


Figure 1.8 Select Installation Folder Page



5. If needed, select **Disk Cost** to view the available space on the selected drive
The *Disk Space* window opens.

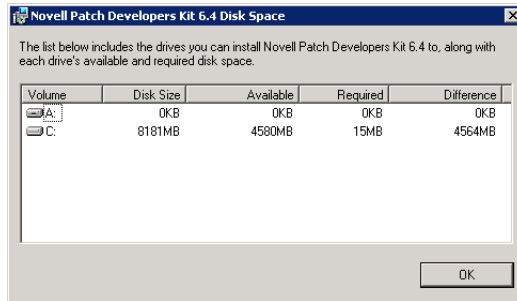


Figure 1.9 Disk Space page

6. Click **OK** to return to the *Select Installation Folder* page
7. Select **Everyone** or **Just Me** to determine user access to the PDK and click **Next**.
The Confirm Installation window opens.

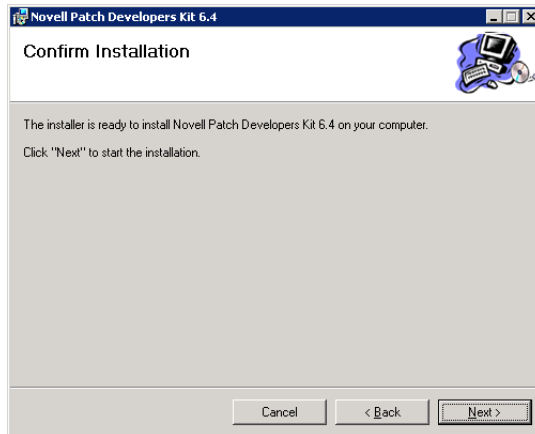


Figure 1.10 Installation Confirmation page



8. Click **Next**.

The PDK installs to the specified location and displays the *Installation Complete* page.

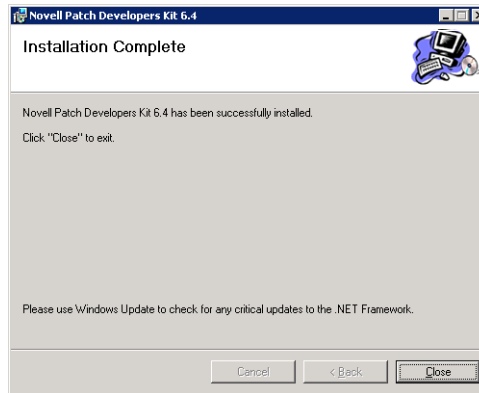


Figure 1.11 Installation Complete page

9. Click **Close** to complete the installation process.

Connecting ZENworks Patch Management Server and the PDK

The PDK will attempt to access the file system on the target Patch Management Server and requires read/write access to that server. If PDK cannot access Patch Management Server, the application will not open.



Note: The PDK requires administrative access to the administrator share of your ZENworks Patch Management Server for copying, adding package files, reading registry entries, and settings.

Accessing the PDK

The following section explains how to start the PDK.

Starting the PDK

1. Select the **Novell Patch Developers Kit**
The *Connect to Server* page opens
2. Type the **Server IP Address** in the **Server Name** field



3. In the **Serial Name or Password** field, type your provided **Serial Number**

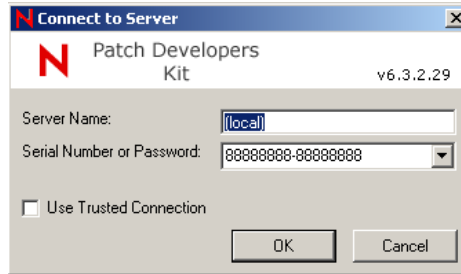


Figure 1.12 Connect to Server page



Note: If you changed the Administrator password on the ZENworks Patch Management Server database, then you will need to use that instead of the serial number

4. Click **OK**
The *Open Vulnerability* screen opens.

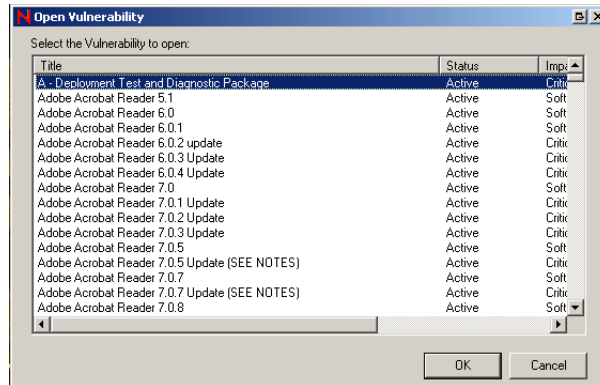


Figure 1.13 Open Vulnerability

Exiting the PDK

1. While in the main window, select **File>Exit**, or click the **Close** button.
The PDK closes.



2 Defining the Properties

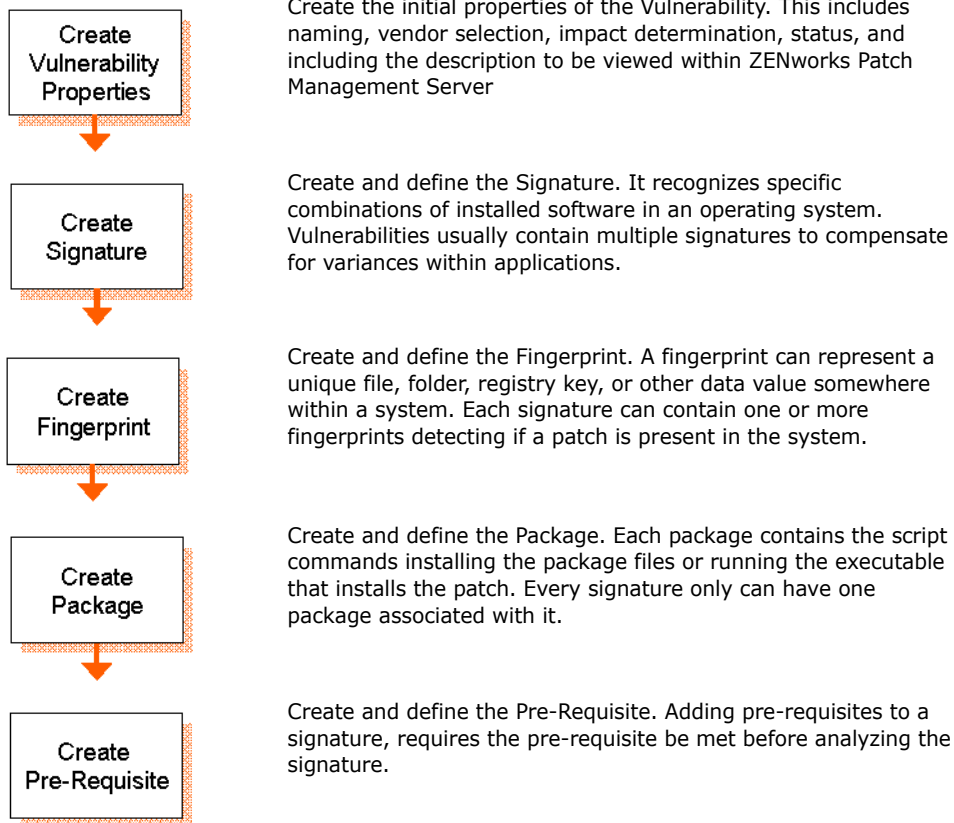
A vulnerability consists of a vulnerability, how it is detected, and its associated patch or patches. It also contains the necessary signatures and fingerprints determining if the vulnerability has been patched.

In This Chapter

- “Building the Vulnerability Process” on page 11
- “Viewing the Vulnerability Properties Window” on page 12
- “Creating and Editing Vulnerabilities” on page 15

Building the Vulnerability Process

The following process details how a vulnerability is created. The process consists of several components - creating the initial properties, and completing the signature details.



Viewing the Vulnerability Properties Window

The *Vulnerability Properties* window allows you to view and/or edit the properties associated with the selected vulnerability.

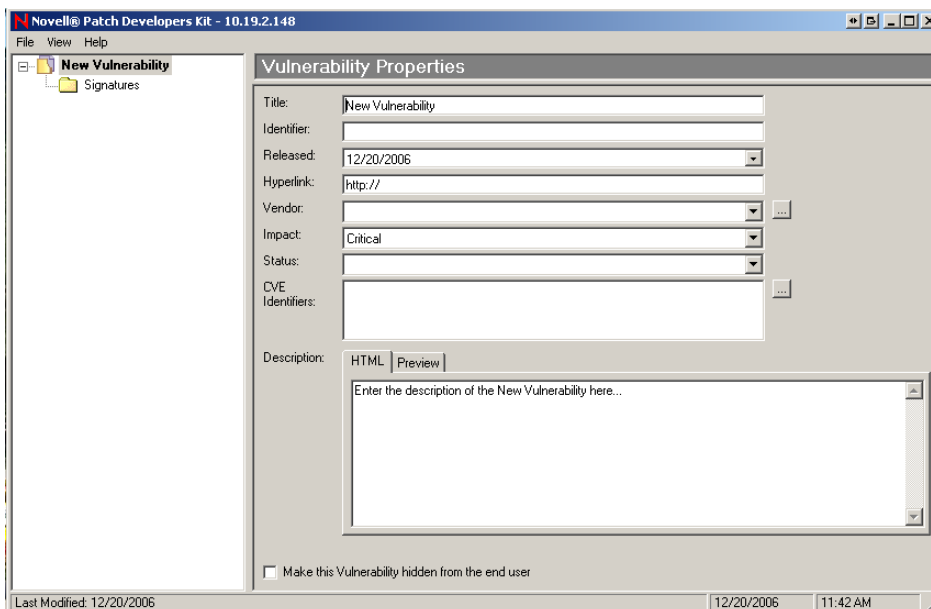


Figure 2.1 Vulnerability Properties window



Tip: When creating a vulnerability, be sure to provide as much information about the patch as possible. To preview the vulnerability description, from within the PDK, click the **Preview** tab.



Vulnerability Properties Window Description

The following table describes the *Vulnerability Properties* window fields.

Table 2.1 Vulnerability Properties

Field	Description
Title	Contains the name of the vulnerability. The vulnerability requires a title to display properly in your ZENworks Patch Management Server
Identifier	Contains the vendor specific number or id value that uniquely correlates with this vulnerability
Released	Contains the date that the vendor released the patch. When creating a new vulnerability, this field is set to the current date by default. This date should be changed to correspond with the date the patch was released by the vendor
Hyperlink	An optional field that provides a link to more information. If a URL is entered in this field, the More Information link in the <i>Vulnerabilities</i> page within Patch Management Server is visible.
Vendor	Represents the company that released the patch. Clicking the drop-down arrow will allow you to select from a list of vendors that are already in the database



Table 2.1 Vulnerability Properties

Field	Description
Impact	<p>Indicates the level of severity for this patch. The values are as follows:</p> <ul style="list-style-type: none"> • Critical - Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. Most of the recent security updates fall in to this category. The patches for this category are automatically downloaded and stored on your ZENworks Patch Management Server. • Critical - 01 - Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. This patch is older than 30 days and has not been superseded. • Critical - 05 - Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. These patches have been superseded. • Critical - Intl - An international patch, where Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. Most of the recent international security updates fall in to this category. After 30 days international patches in this category will be moved to Critical - 01. • Detection - These vulnerabilities contain signatures that are common to multiple vulnerabilities. They contain no associated patches and are only used in the detection process. • Informational - These vulnerabilities detect a condition that Novell or the product manufacturer has determined as informational. If the report has an associated package, you may want to install it at your discretion. • Recommended - Novell or the product manufacturer has determined that this patch, while not critical or security related is useful and should be applied to maintain the health of your computers. • Software - These vulnerabilities are software applications. Typically, this includes software installers. The vulnerabilities will show not patched if the application has not been installed on a machine. • Task - This category contains tasks which administrators may use to run various detection or deployment tasks across their network. • Virus Removal - This category contains packages which administrators may use to run various virus detections across their network. Anti-Virus tools and updates are included in this category.
Status	<p>Defines the status of the vulnerability. If set to Active, Patch Management Server users will be able to view this vulnerability in the <i>Vulnerabilities</i> page. If set to Beta, only Patch Management Server sites set for Beta use will be able to view the vulnerability</p>
CVE Identifier	<p>Allows for the patch to be defined and classified using the Common Vulnerabilities and Exposures standard. See http://cve.mitre.org for more information.</p>
Description	<p>Contains a text description of the vulnerability. The information is displayed in the <i>Vulnerabilities</i> page of ZENworks Patch Management Server also.</p>



Creating and Editing Vulnerabilities

The following section describes the process for creating and editing Vulnerabilities. This process creates a new Vulnerability and saves it to ZENworks Patch Management Server.

Creating a Vulnerability involves several procedures to complete the patch, and comprises of two major components: creating the initial properties, and completing the details. The following process details the basic process, and directs you to the more involved procedures for that component.

To Create the Properties for a Vulnerability

1. Launch the PDK
The *Open Vulnerability* window opens.
2. Click **Cancel**
The *Vulnerability Properties* form opens.

The screenshot shows the 'Vulnerability Properties' dialog box within the Novell Patch Developers Kit. The window title is 'Novell Patch Developers Kit - 10.19.2.148'. The dialog has a menu bar with 'File', 'View', and 'Help'. On the left, there is a tree view with 'New Vulnerability' selected, and a sub-item 'Signatures'. The main area contains the following fields:

- Title: [Text input field]
- Identifier: [Text input field]
- Released: [Date dropdown menu, showing 12/20/2006]
- Hyperlink: [Text input field, showing http://]
- Vendor: [Dropdown menu]
- Impact: [Dropdown menu, showing Critical]
- Status: [Dropdown menu]
- CVE Identifiers: [Text input field]
- Description: [HTML | Preview | Text area with placeholder 'Enter the description of the New Vulnerability here...']

At the bottom, there is a checkbox labeled 'Make this Vulnerability hidden from the end user'. The status bar at the bottom shows 'Last Modified: 12/20/2006', '12/20/2006', and '11:48 AM'.

Figure 2.2 Vulnerability Properties

3. In the **Title** field, type a name for the Vulnerability
4. In the **Identifier** field, type a unique identifier for the Vulnerability. This can be determined by the individual user or can be the Vendor's
5. In the **Released** field, select the date the patch is to be released. Use the Vendor's date if required.
6. In the **Hyperlink** field, type the URL for the vendor



7. In the **Vendor** field, select a Vendor.



Note: To add, edit or delete a vendor, refer to “Working with Vendors” on page 16.

8. In the **Impact** field, select the Impact from the drop-down list
9. In the **Status** field, select the Vulnerability status from the drop-down list
10. Type or select a CVE Identifier in the **CVE Identifier** fields
11. In the **Description** field, type the description that will be viewed within ZENworks Patch Management Server

Working with Vendors

To manage vendors open the *Vendor Management* window by selecting **Vendor Management** from the **View** menu. From within the *Vendor Management* window you can add, edit, or delete vendors.



Note: Vendors created by PatchLink cannot be edited or deleted from the Novell PDK.

To Add New Vendors

1. Select **View > Vendor Management**
The *Vendor Management* window opens

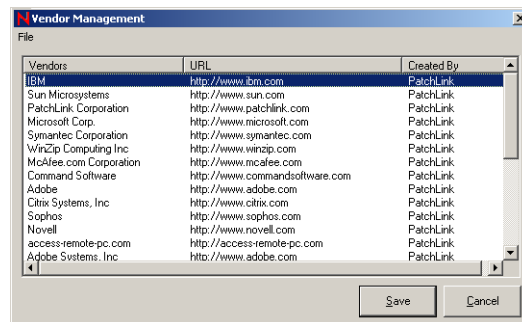


Figure 2.3 Vendor Management



2. Select **File > New Vendor...**
The *Add Vendor* window opens

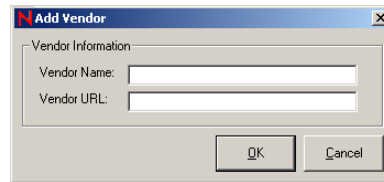


Figure 2.4 Add Vendor

3. In the **Vendor Name** field, type the Vendor Name
4. In the **Vendor URL** field, type the web address of the vendor
5. Click **OK**
The PDK adds the the vendor to the database in the *Vendor Management* window.
6. Click **Save**
The PDK saves the changes

To Edit Vendors

1. Select **View > Vendor Management**
The *Vendor Management* window opens
1. Select **File > Edit Vendor...**
The *Edit Vendor* window opens
2. Make any changes to the **Vendor Name** and **Vendor URL**
3. Click **OK**
The database saves the changes and closes the *Edit Vendor* window



Note: Only Vendors you create can be edited.

4. Click **Save**
The changes are saved to the Vendor and the *Vendor Management* window closes.



To Delete Vendors

1. Select **View > Vendor Management**
The *Vendor Management* window opens
2. Select the vendor you want to remove from the list
3. Select **File > Delete Vendor**
The vendor is removed from the database
4. Click **Save**
The PDK saves the changes and closes the *Vendor Management* window



Note: Only Vendors you created can be deleted.

- 5.



3 Working with Signatures

Each signature is used to recognize a specific operating system and/or a combination of installed software applications and services. If there are multiple unique configurations that must be recognized, the vulnerability will contain multiple signatures. Likewise, if a patch requires unique installation files for each operating system, it will contain multiple signatures.

In This Chapter

- “Defining Signatures and Pre-Requisite Signatures” on page 19
- “Adding a Signature” on page 21
- “Editing a Signature” on page 23
- “Removing a Signature” on page 23

Defining Signatures and Pre-Requisite Signatures

From the *Signature Summary* window you may add, edit, or delete signatures from the vulnerability.

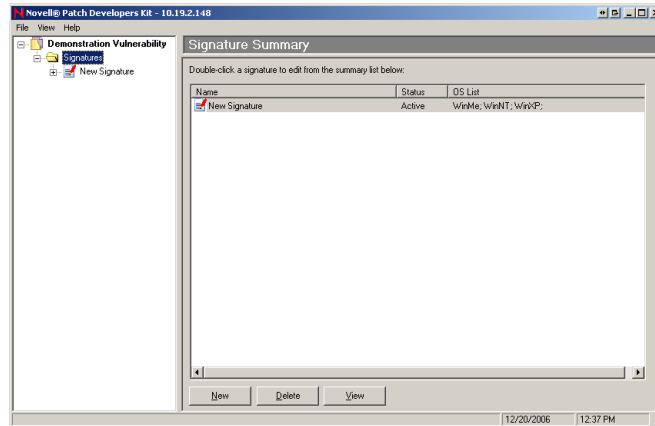


Figure 3.1 Signature Summary

Table 3.1 Signature Summary buttons

Button	Description
New	Creates and adds a new signature to the <i>Signature Summary</i>
Delete	Removes the currently selected signature
View	Opens the <i>Signature Properties</i> window



Defining the Signature and Pre-Requisite Signature Properties

The *Signature Properties* window contains the signature's title, status, and applicable operating systems.

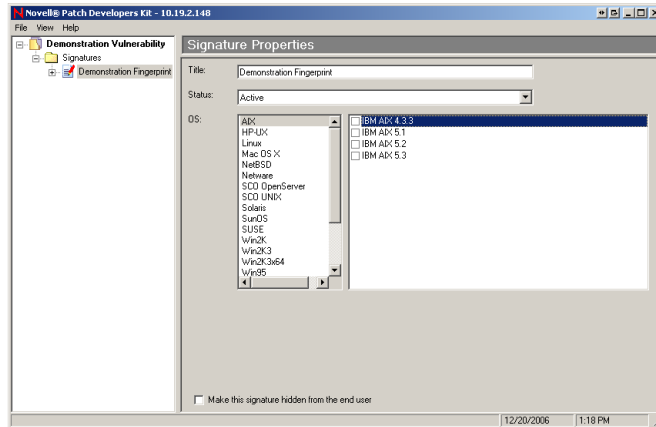


Figure 3.2 Signature Properties

Table 3.2 Signature Properties fields

Field	Description
Title	The text identifier for the signature. This title is not visible in the Patch Management Server web interface
Status	Where you define whether the signature is for regular or beta use. For regular use, you must set the status to <i>Active</i> , allowing Patch Management Server users to see the package on the <i>Packages</i> page. For beta use, set the status to <i>Beta</i> , limiting use to only Patch Management Server beta sites
OS	Where you define the operating systems that the signature applies to. When creating a signature it is important that you select only the operating systems which are applicable to the signature



Working With Signatures

The following procedure describes the signature defining process. The signature window is to the left of the *Vulnerability Properties* window. For every signature, you can have multiple fingerprints, packages, and pre-requisites.

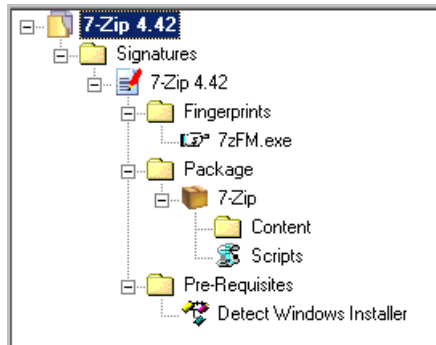


Figure 3.3 Signatures Window

Adding a Signature

The following procedure describes the steps for adding a signature to a new Vulnerability.

To Add a New Signature

1. Select the *Signature* folder in the Details area
The *Signature Summary* window opens



2. Click **New**
The PDK creates a blank Signature

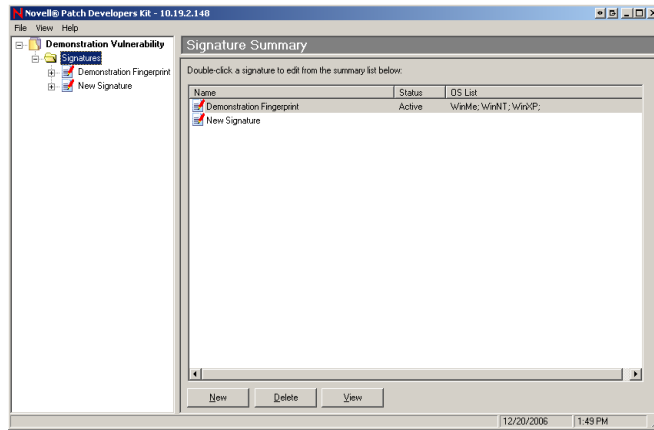


Figure 3.4 Signature Summary

3. Highlight the signature, and click **View**
The *Signature Properties* window opens

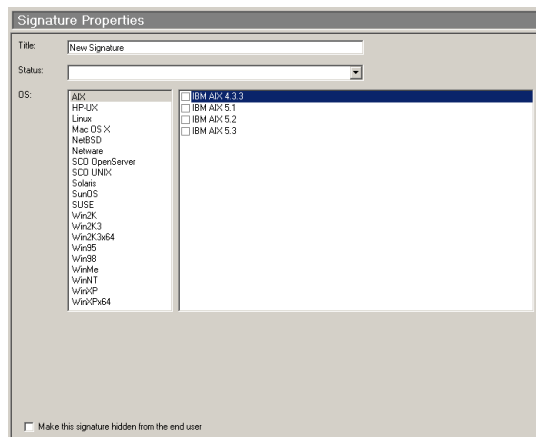


Figure 3.5 Signature Properties

4. In the **Title** field, type a name for the signature. This title will not be seen in ZENworks Patch Management Server.
5. In the **Status** field, select **Active** or **Beta** from the drop-down list.



- **Active** - ZENworks Patch Management Server users will be able to see the package on the *Packages* page.
 - **Beta** - Only sites set up for Beta use will be able to view the signature
6. In the **OS list**, select the operating system applicable to the signature.
The versions of the selected operating system display in the OS list details field.
 7. Select the version of the operating system for the current signature.

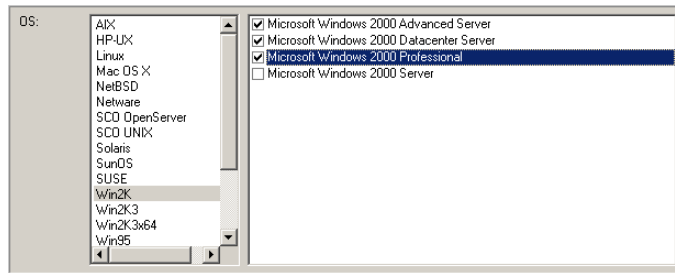


Figure 3.6 OS list details



Note: Double-clicking the *Operating System* selects all *versions* of that OS.

8. If needed, select **Make this signature hidden from the end user**
9. Expand the **New Signature** folder to continue to **Fingerprints**

Editing a Signature

To Edit a Signature

1. Select the signature you want to edit
2. Click the **View** button on the bottom of the *Signature Summary* window
3. Modify the title, status, and/or operating system settings as needed
4. Select the title of the Vulnerability to return to the *Vulnerability Properties* window

Removing a Signature

To Remove a Signature

1. Select the signature you want to delete from within the *Signature Summary* window
2. Click **Delete**
3. Click **OK** to confirm the deletion





4 Creating Fingerprints

Within a signature there can be several fingerprints. The fingerprint detects if a patch is present within a device. It can be a file, directory, registry key, or a value within a registry.

In This Chapter

- “Viewing the Fingerprint Summary Window” on page 25
- “Adding Fingerprints” on page 27
- “Removing a Fingerprint” on page 30
- “Creating Fingerprints in Expert Mode (XML)” on page 30
- “Using Fingerprint Types” on page 31

Viewing the Fingerprint Summary Window

The *Fingerprint Summary* window allows you to display all fingerprints associated with a signature. To display the *Fingerprint Summary* window click on the appropriate signatures *Fingerprints* folder.

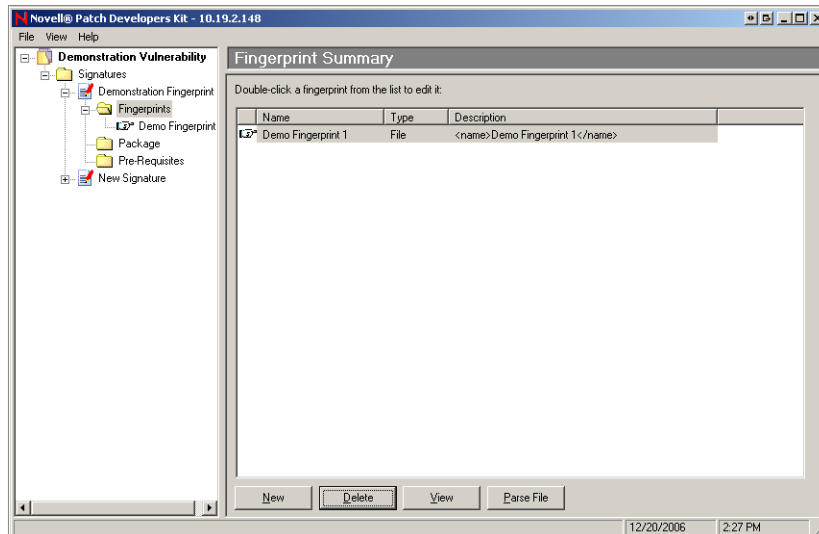


Figure 4.1 Fingerprint Summary



The Fingerprint Summary Buttons allow for creating, viewing, deleting, or creating multiple fingerprints.

Table 4.1 Fingerprint Summary Buttons

Button	Description
New	Creates and adds a new fingerprint to the Fingerprint Summary
Delete	Deletes a fingerprint from the Vulnerability
View	Opens the <i>Fingerprint Properties</i> window
Parse File	Allows for adding multiple fingerprints using an XML text file

Fingerprint Types

There are seven types of fingerprints available. The table provides a brief definition of each option. The following sections provide field definitions for each fingerprint screen option.

Table 4.2 Fingerprint Type and Description

Fingerprint	Used to	Target OS
File	Determine presence and properties of files and directories	Windows UNIX/Linux/Mac
Registry	Extract data from the Windows Registry	Windows
WMI	Detect information about a system's operating system, name, distribution, or version	Windows UNIX/Linux/Mac
Systeminfo	Retrieve information about a device. Including the OS Name and/or version, and Architecture	Windows UNIX/Linux/Mac
Expression	Computes logical operations based on presence or absence of other fingerprints	UNIX/Linux/Mac
Patch	Determine presence of special components such as patches	UNIX/Linux/Mac
Script	Allows for custom XML script creation of fingerprints	Windows UNIX/Linux/Mac



Adding Fingerprints

Single or multiple fingerprints can be added to the vulnerability.

To Add a New Fingerprint

1. In the *Fingerprint Summary* window, click **New**
A **New Fingerprint** is added to the *Fingerprint* directory

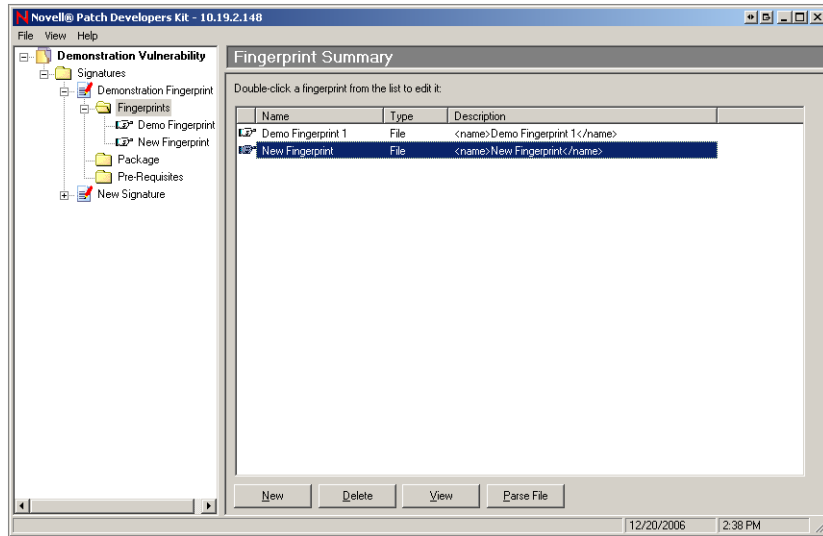


Figure 4.2 Fingerprint Summary



2. Click **View**

The *Fingerprint Properties* window opens. The *File* fingerprint is the default view.

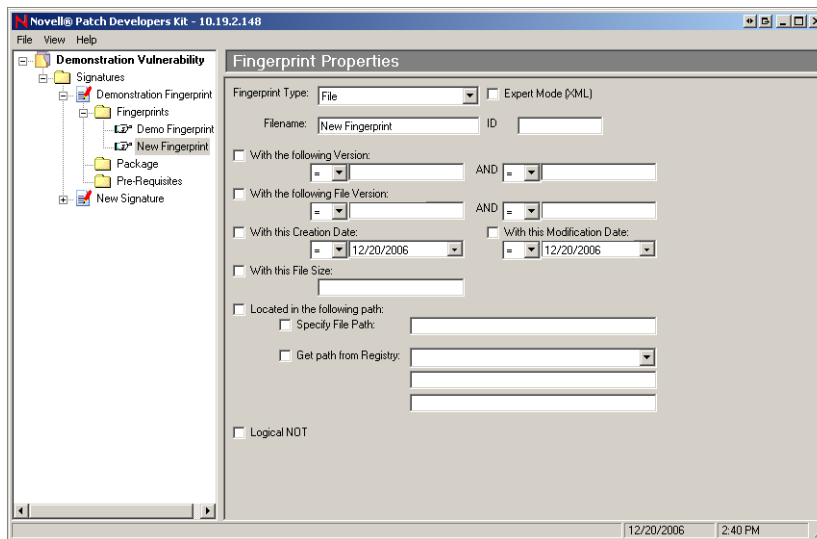


Figure 4.3 Fingerprint Properties - File Fingerprint

3. Define the appropriate fingerprint properties

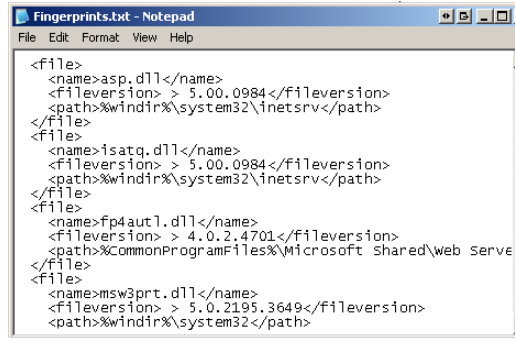
Note: For detailed information on each fingerprint, field definitions, and usage suggestions, see “[Using Fingerprint Types](#)” on page 31.



Adding Multiple Fingerprints

The **Parse File** allows for creating multiple fingerprints using an XML text document.

To create the XML text file for parsing, write the XML similar to writing a fingerprint using *Expert XML* mode. Refer to “[Creating Fingerprints in Expert Mode \(XML\)](#)” on page 30 for more information.



```

<file>
<name>asp.dll</name>
<fileversion> > 5.00.0984</fileversion>
<path>%windir%\system32\inetrv</path>
</file>
<file>
<name>isatq.dll</name>
<fileversion> > 5.00.0984</fileversion>
<path>%windir%\system32\inetrv</path>
</file>
<file>
<name>fp4aut1.dll</name>
<fileversion> > 4.0.2.4701</fileversion>
<path>%CommonProgramFiles%\Microsoft Shared\Web Serve
</file>
</files>
<file>
<name>msw3prt.dll</name>
<fileversion> > 5.0.2195.3649</fileversion>
<path>%windir%\system32</path>

```

Figure 4.4 XML text file example

To Use the Parse File Option

1. In the *Fingerprint Summary* window, click **Parse File**
The *Open File* window opens

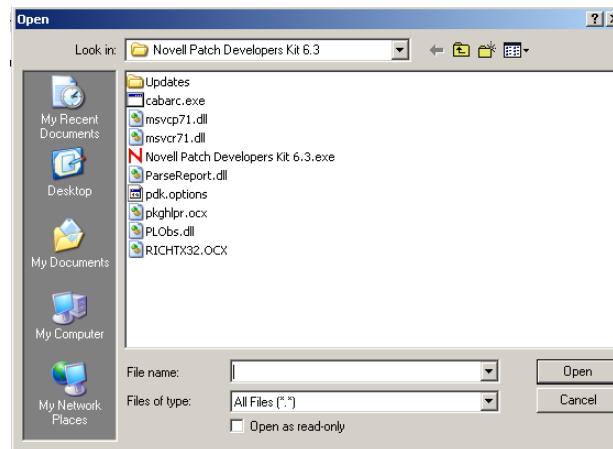


Figure 4.5 Save file



2. Select the XML text file you have written
3. Click **Open**
4. Click **OK** to load the file
The fingerprints display in the *Fingerprint Summary* window

Removing a Fingerprint

1. Select the fingerprint that you want to remove
2. Click **Delete**
The *Delete Confirmation* window opens
3. Click **OK** to remove the fingerprint from the vulnerability report

Creating Fingerprints in Expert Mode (XML)

The Expert Mode (XML) Checkbox is provided for creating fingerprints using XML.

To Switch to Expert Mode

1. In the *Fingerprint Properties* window, select the **Fingerprint** needed.
The selected Fingerprint displays in the *Fingerprint Properties* window
2. Select the **Expert Mode (XML)** checkbox
The window displays the *Expert Mode* field

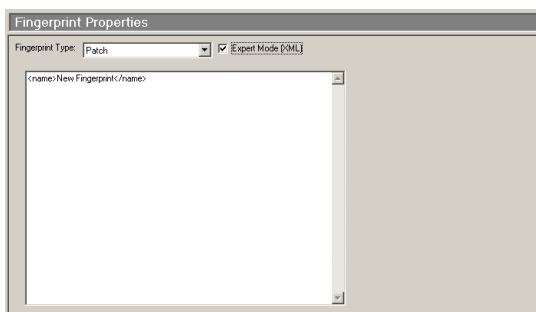


Figure 4.6 Expert Mode (XML) Example

3. Type the XML fingerprint code
4. Create another fingerprint, or continue to Packages



Tip: To create multiple fingerprints using XML, use the Parse File option. See “[Adding Multiple Fingerprints](#)” on page 29 for more information.

Using Fingerprint Types

The following section describes each fingerprint option within the *Fingerprint Properties* window. The available properties vary depending upon the fingerprint type. The following fingerprints are available:

- “Using the File Fingerprint”
- “Using the Registry Fingerprint”
- “Using the WMI Fingerprint”
- “Using the SystemInfo Fingerprint”
- “Using the Expression Fingerprint”
- “Using the Patch Fingerprint”
- “Using the Script Fingerprint”



Note: See “[Adding Fingerprints](#)” on page 27 for steps on how to create fingerprints.



Using the File Fingerprint

The *File Fingerprint* determines the presence and properties of files and directories within Windows UNIX/Linux/Mac.

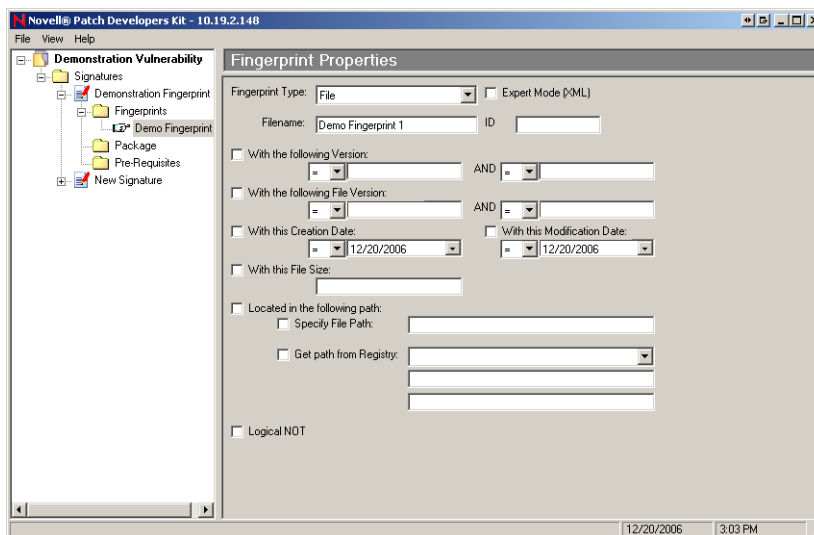


Figure 4.7 Fingerprint Properties

The following table describes the fields, how to use them, and their equivalent XML tag used for Expert mode. Not all fields are required.

Table 4.3 File Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Fingerprint Type	Identifies the type of fingerprint	Select the fingerprint needed for the report	N/A
Expert Mode (XML)	Allows for entering fingerprint data in XML	Toggle between the fingerprint property fields and a text field, allowing you to add/view the properties using XML	N/A
Filename	Enter a specific filename	<ul style="list-style-type: none"> Specify an environment variable that includes either the filename or the path and filename. Leave the field blank if you are looking for the existence of a directory 	<name>
ID	Patch identifier that can be customized by company	The number can be specific to a company or department. Use for integrating with the Expression fingerprint.	N/A



Table 4.3 File Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
With the following Version	Search for a minimum version, or a range of versions	<ul style="list-style-type: none"> Enter a specific version number (=) Search for a version less than or equal to (<=) or greater than or equal to (>=) a specific value Search for a version within a range of values 	<version>
AND	Allows for additional qualifiers	Add additional parameters. Both values must be true.	N/A
With the following File Version	Searches for either a minimum file version, or a range of file versions	<ul style="list-style-type: none"> Enter a specific file version number (=) Search for a file version less than or equal to (<=) or greater than or equal to (>=) a specific value Search for file versions within a range of values 	<fileversion>
With this Creation Date	Allows you to search based upon the creation date	<ul style="list-style-type: none"> Enter a specific creation date Search for a creation date less than or equal to (<=) or greater than or equal to (>=) a specific date 	<created>
With this File Size	Searches for a file based upon an exact file size (in bytes)	Can only be used to search for a specific size.	<size>
With this Modification Date	Allows for locating a patch with a specific modification date range	<ul style="list-style-type: none"> Enter a specific modification date Search for a modification date less than or equal to (<=) or greater than or equal to (>=) a specific date 	<modified>
Located in the following path - Specify File Path	Allows you to specify a relative path to be used when looking for a file	<p>Specify an environment variable containing a path or a path/filename</p> <ul style="list-style-type: none"> If the environment variable has a filename included, and a filename was not specified in the Filename field, the filename returned from the variable is used If the environment variable has a filename included, and a file was specified in the Filename field, the filename returned by the variable will be discarded <p>Specify a relative path. Will search all local drives and look for a path that ends with the selected parameter.</p> <p>Leave blank for a broad search for the file (not recommended since this will search every drive)</p> <p>Enter an absolute path such as C:\winnt\system32 (not recommended because of the ability to customize the installation path of an application)</p>	<path>



Table 4.3 File Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Located in the following path - Get Path from Registry	Retrieves the path from the registry	<p>The root for the entry is selected from the drop-down list. Type the KEY and VALUE into the respective fields. The available ROOT values are:</p> <ul style="list-style-type: none"> • HKEY_LOCAL_MACHINE • HKEY_CLASSES_ROOT • HKEY_CURRENT_USER • HKEY_USERS • HKEY_CURRENT_CONFIG <p>Manually enter the KEY</p> <p>For the VALUE field, you can either:</p> <ul style="list-style-type: none"> • Manually enter the VALUE • Enter value of (Default) to use KEY's default value 	<pre><root> <key> <value></pre>
LogicalNOT	Changes search from checking if a file exists to checking if it does NOT exist	Use for confirming if a file or directory was previously created.	<pre><not></pre>

Fingerprint File XML Example

The following example consists of an XML script that includes all the possible XML parameters.

```
<File>
<name>outlook.exe</name>
<version> > 4.01.2345b </version>
<version> < 5.00.2789 </version>
<fileversion> > 5.01.2345 </fileversion>
<Created> > 5/30/2001 12:01:04 PM </Created>
<modified> > 5/30/2001 12:01:04 PM </modified>
<size>4252</size>
<root>HKEY_LOCAL_MACHINE</root>
      <key>SOFTWARE\Classes\Software\Adobe\Exe</key>
      <value>(Default)</value>
</File>
```



LogicalNOT XML Example Script

The following example includes the script using the LogicalNot option.

```
<File>
  <name>temptest.txt</name>
  <path>%WINDIR%\temp</path>
  <not>1</not>
</File>
```

Using the Registry Fingerprint

Used to extract data from the windows registry. Only works on Windows operating systems.

Figure 4.8 Fingerprint Properties - Registry

The following table describes the fields, how to use them, and their equivalent XML tag used for Expert mode.

Table 4.4 Registry Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Fingerprint Type	Identifies the type of fingerprint	Select the fingerprint needed for the report	N/A
Expert Mode (XML)	Allows for entering fingerprint data in XML	Toggle between the fingerprint property fields and a text field, allowing you to add/view the properties using XML	N/A



Table 4.4 Registry Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Root Key	Registry root key	Searches for the root key. You must also specify the SubKey and Value Name . The following are possible values for Root Key: <ul style="list-style-type: none"> • HKEY_CLASSES_ROOT • HKEY_CURRENT_USER • HKEY_LOCAL_MACHINE • HKEY_USERS • HKEY_CURRENT_CONFIG 	<root>
SubKey	Registry subkey	Searches for registry subkey. When using the registry fingerprint type you must specify a SubKey	<key>
Value Name	Value of attribute	Defines the registry value within the specified key that will be searched for. You can either enter a specific value or use the default key by entering (Default)	<value>
With a value that matches	Allows for a search of a matching value.	Enter a specific value and add additional parameters.	N/A
LogicalNOT	Changes search from checking if a registry exists to checking if it does NOT exist	Use for confirming if a registry was previously created.	<not>

Registry XML Example

The following example consists of an XML script that includes all the possible XML parameters.

```
<Registry>
<root>HKEY_LOCAL_MACHINE</root>
      <key>SOFTWARE\Classes\Software\Adobe\Exe</key>
      <value>(Default)</value>
</Registry>
```

LogicalNOT XML Example Script

The following example includes the script using the LogicalNot option.

```
<Registry>
  <name>temptest.txt</name>
  <path>%WINDIR%\temp</path>
  <not>1</not>
</Registry>
```



Using the WMI Fingerprint

Used to detect information about a system such as operating system name, distribution, or version. Works on both Windows or UNIX/Linux operating systems.

Figure 4.9 Fingerprint Properties - WMI

The following table describes the fields, how to use them, and their equivalent XML tag used for Expert mode. Not all fields are required.

Table 4.5 WMI Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Fingerprint Type	Identifies the type of fingerprint	Select the fingerprint needed for the report	N/A
Expert Mode (XML)	Allows for entering fingerprint data in XML	Toggle between the fingerprint property fields and a text field, allowing you to add/view the properties using XML	N/A
Name	Name of script	Contains the name of the script	<name>
Expression ID	Expression ID	Used to give this fingerprint an expression ID so that it can be used in a more complicated expression (UNIX/Linux vulnerabilities only)	<eid>
Content	Content of Script	Contains the content of the script to be run	<content>



WMI XML Example

The following example consists of an XML script that includes all the possible XML parameters.

```
<WMI>
<name>LPRng requirements script</name>
<content>
<!--this is an example script-->
  #!/bin/sh
  if [ -f myfile.txt ]; then
    echo "File Detected"
  else
    echo "File Not Found"
  exit 0
</content>
<eid>c7</eid>
</WMI>
```

Using the SystemInfo Fingerprint

Used to retrieve information about a computer such as operating system name, architecture, and operating system version. Works on both Windows and UNIX/Linux operating systems.

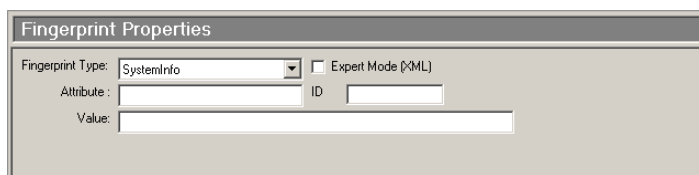


Figure 4.10 Fingerprint Properties - System Info

The following table describes the fields, how to use them, and their equivalent XML tag used for Expert mode. Not all fields are required.

Table 4.6 Systeminfo Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Fingerprint Type	Identifies the type of fingerprint	Select the fingerprint needed for the report	N/A
Expert Mode (XML)	Allows for entering fingerprint data in XML	Toggle between the fingerprint property fields and a text field, allowing you to add/view the properties using XML	N/A



Table 4.6 Systeminfo Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Attribute	Script name	This field allows you to specify which system attribute is being determined. The attributes are: <ul style="list-style-type: none"> • Architecture • AgentVersion • OSName • OSDistribution • OSVersion • OSKernelVersion 	<systemattribute>
Value	Value of attribute	Allows you to specify which value the attribute will be compared against	<value>
ID	Expression ID	Used to give this fingerprint an expression ID so that it can be used in a more complicated expression (UNIX/Linux vulnerabilities only)	<eid>

SystemInfo XML Example

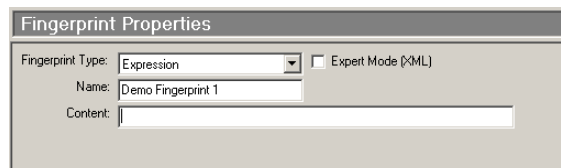
The following example consists of an XML script that includes all the possible XML parameters.

This example determines whether the architecture of the client machine is an iX86 greater than or equal to i386. Then the result is put in the variable c0, which can then be used in a logical expression to determine if a signature is present.

```
<SystemInfo>
  <systemattribute>Architecture</systemattribute>
  <value>_GE_i386</value>
  <eid>c0</eid>
</SystemInfo>
```

Using the Expression Fingerprint

The expression fingerprint type computes logical operations based on the presence or absence of other fingerprints. Only works on UNIX/Linux operating systems.

**Figure 4.11** Fingerprint Properties - Expression

The following table describes the fields, how to use them, and their equivalent XML tag used for Expert mode. Not all fields are required.

Table 4.7 Expression Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Fingerprint Type	Identifies the type of fingerprint	Select the fingerprint needed for the report	N/A
Expert Mode (XML)	Allows for entering fingerprint data in XML	Toggle between the fingerprint property fields and a text field, allowing you to add/view the properties using XML	N/A
Name	Name of script	Contains the name of the script	<name>
Content	Content of script	Contains the content of the script to be evaluated	<content>



Note: Prior to using a term in an expression it must be already defined by setting the ID (or Entity ID) of the other components.

Expression XML Example

The following example consists of an XML script that includes all the possible XML parameters. This example determines whether the client machine has

1. Any version of the application called KDE
2. A new agent
3. An iX86 architecture of at least i386

This was accomplished by associated other fingerprint types to the cX variables that are in the logical expression.

In this example:

- c0 is the result of attempting to detect a new agent
- c1 is the result of attempting to detect an iX86 architecture \geq to i386
- c2 through c5 are used to detect the presence of the KDE application (c0 and c1 must be present)
- c2 through c4 are the result of attempting to detect components, any one of which must be present in the KDE application. (i.e. either c2 OR c3 OR...c4 must be present)



- c5 is the result of attempting to detect components, each of which must be present in the KDE application

```
<Expression>
<name>Any kde with new agent and -GE- i386</name>
<content>c0 AND c1 AND (c2 | c3 | c4 ) AND c5</content>
</Expression>
```

Using the Patch Fingerprint

Used to determine the presence of special components such as patches. In the case of *Red Hat Linux* they are the `rpms` while in case of *Solaris* they are the `SUNW` packages. Only works on UNIX/Linux operating systems.

Figure 4.12 Fingerprint Properties - Patch

The following table describes the fields, how to use them, and their equivalent XML tag used for Expert mode. Not all fields are required.

Table 4.8 Patch Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Fingerprint Type	Identifies the type of fingerprint	Select the fingerprint needed for the report	N/A
Expert Mode (XML)	Allows for entering fingerprint data in XML	Toggle between the fingerprint property fields and a text field, allowing you to add/view the properties using XML	N/A
Name	Name of package	Contains the name of the rpm or Solaris package that you are searching for	<name>
Version	Package version	The version against which the package will be compared	<version>



Table 4.8 Patch Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Release	Release version	The release against which the package will be compared	<release>
Expression ID	Expression ID	Used to give this fingerprint an expression ID so that it can be used in a more complicated expression (UNIX/Linux vulnerabilities only)	<eid>

Patch XML Example

The following example consists of an XML script that includes all the possible XML parameters. This example determines whether a patch with the *Solaris Patch ID* of 106468-05 exists on a *Solaris* computer

```
<name>106468-05</name>
<version>106468</version>
<release>_GE_05</release>
<eid>c0</eid>
```

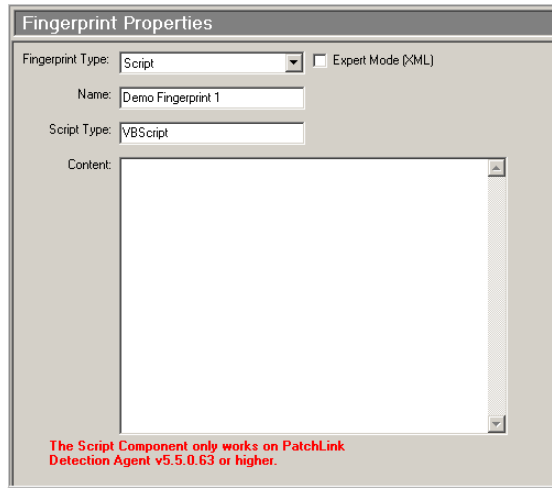
Using the Script Fingerprint

The Script Fingerprint can be used to detect information about a computer such as operating system name, version, services, and other values using *SQL like* queries. Works on both Windows and UNIX/Linux operating systems.



Warning: The Script component will only work on ZENworks Patch Management Agents which are version of 5.5.0.63 or higher





The Script Component only works on PatchLink Detection Agent v5.5.0.63 or higher.

Figure 4.13 Fingerprint Properties - Scripts

The following table describes the fields, how to use them, and their equivalent XML tag used for Expert mode. Not all fields are required.

Table 4.9 Script Fingerprint Fields

Field	Description	Usage Suggestions	XML Tag
Fingerprint Type	Identifies the type of fingerprint	Select the fingerprint needed for the report	N/A
Expert Mode (XML)	Allows for entering fingerprint data in XML	Toggle between the fingerprint property fields and a text field, allowing you to add/view the properties using XML	N/A
Name	Script name	Contains the name of the script	<name>
Script Type	Type of script	Contains the type of script to be executed. Currently the PDK only supports the VBScript script type	<type>
Content	Contents of scrip	Type the actual script to be used. The VBScript must reference the PLCCAgent and return a value of either TRUE or FALSE.	<content>



Script XML Example

The following example consists of an XML script that includes all the possible XML parameters.

```
<name>Check SQL</name>
<type>VBScript</type>
<contents>
<!--this is an example script-->
    Dim fso
    set fso = CreateObject("scripting.FileSystemObject")
    If fso.FileExists("MYFile.exe") = TRUE Then
        PLCCAgent.SetReturnCode "1", "Success"
    Else
        PLCCAgent.SetReturnCode "0", "Failed"
    End If
</contents>
```



5 Working With Packages

The Package includes the executable files, compress data files, and the scripts required to install the application or patch.

In This Chapter

- “Defining Packages” on page 45
- “Defining Package Content” on page 49
- “Adding a New Package” on page 51
- “Removing a Package” on page 53
- “Editing a Package” on page 54
- “Adding Files to a Package” on page 55

Defining Packages

The *Package Summary* screen allows you to view and define the package options for the Vulnerability.

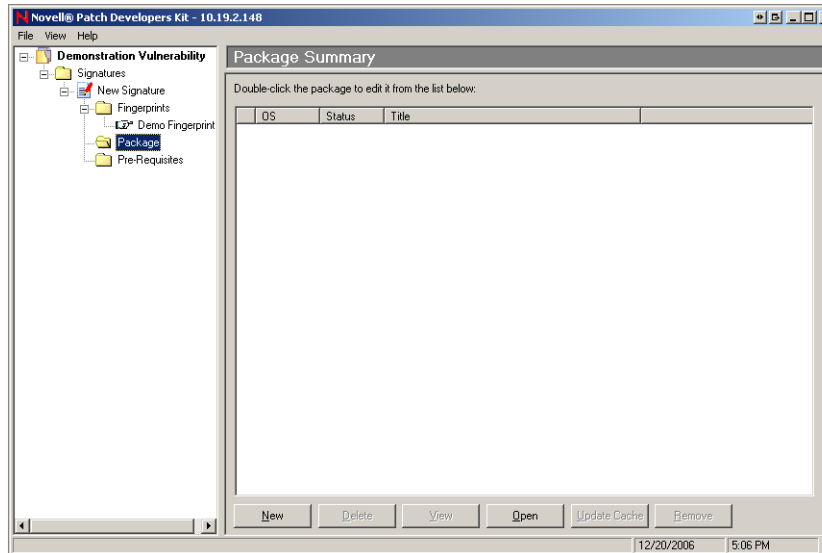


Figure 5.1 Package Summary



The following table describes the Package Summary buttons and their functions.

Table 5.1 Package Summary buttons

Button	Description
New	Creates and adds a new package to the signature
Delete	Deletes the currently selected package from this vulnerability, all other vulnerabilities, and the Patch Management Server
View	Opens the <i>Package Properties</i> window
Open	Adds an existing package to the signature. You can only add one existing package to a signature.
Update Cache	Downloads a new copy of the package to the Patch Management Server
Remove	Removes the currently selected package from this vulnerability



Warning: The **Delete** button will permanently delete the package from this vulnerability and all other vulnerabilities. The package can only be recovered if you have previously made a backup (outside of Patch Management Server) of the package.



Viewing the Package Properties

The *Package Properties* window defines the applicable operating systems, behavior, and description of the package.

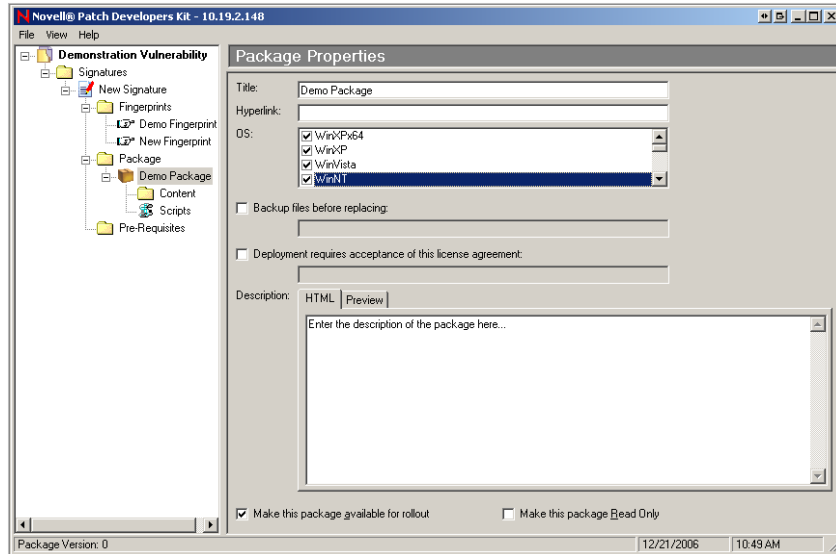


Figure 5.2 Package Properties

The following table describes the Package property fields and their functions.

Table 5.2 Package Properties Field

Field	Description
Title	Displays the package name, usually the same as its associated signature
Hyperlink	References a web page which will further define the package
OS	Defines which Operating System(s) this package is applicable
Backup Files Before Replacing	Defines the backup directory and enables the PDK to archive the files.
Deployment requires acceptance of this license agreement	Requires the license and license agreement that must be displayed, and accepted, prior to deployment of this package



Table 5.2 Package Properties Field

Field	Description
Description	Contains a brief description of the package contents. The description field also contains Novell deployment flags that are interpreted as options within the deployment wizard.
Make this package available for rollout	Defines whether the package is available for deployment. If this checkbox is not selected, the package will not be deployed.



Defining Package Content

The *Package Content* window displays the files and directories included in the package. To display the package contents.

Package Content			
Name	Size	Type	Modified
📁 %TEMP%	0	File Folder	10/28/2006 11:01:45 AM

Figure 5.3 Package Content

Working with Package Scripts

The *Package Scripts* window displays the **Pre-Script**, **Command Line Script**, and **Post-Scripts** when applicable.

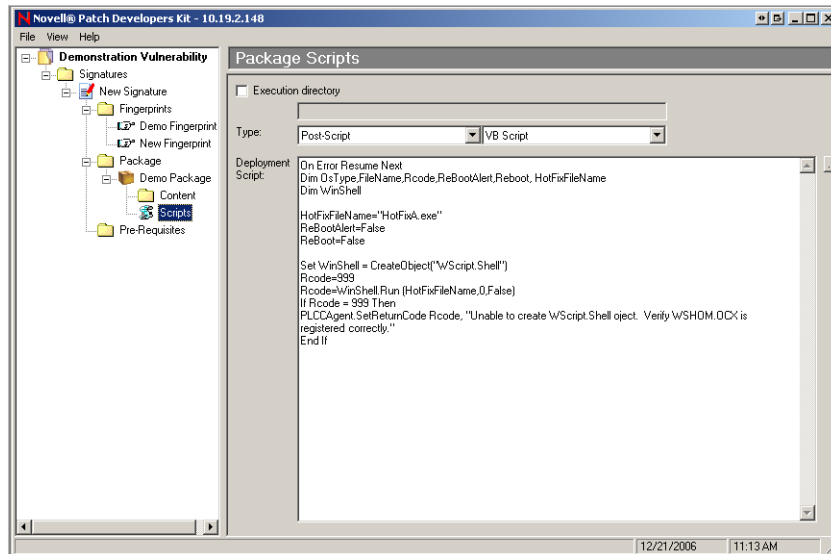


Figure 5.4 Package Properties



The following table describes the Package Properties fields and descriptions.

Table 5.3 Package Properties fields

Field	Description
Execution Directory	Displays the directory under which the selected script will execute, preventing the need for full paths in the script
Type	Defines the script <ul style="list-style-type: none">• Pre Script - used to test for a condition of the machine• Command Line Script - used to launch executables. The format is the same as a standard CMD or BAT file• Post Script - used for operations, deleting files, starting services, or installers. Can take the form of VBScript or JScript.• Microsoft VB Script - language option for the script• Microsoft Jscript - language option for the script• .BAT file - running command from the command prompt
Deployment Script	Contains the actual script to be executed



Adding a New Package

To Add a New Package

1. Click **New**

A *New Package* is added to the *Package Summary* window

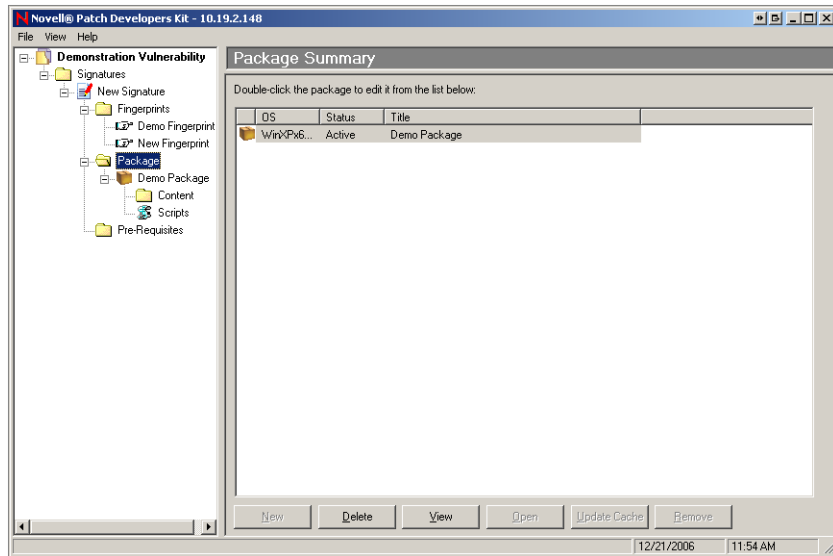


Figure 5.5 Package Summary



2. Click **View**
The *Package Properties* window opens

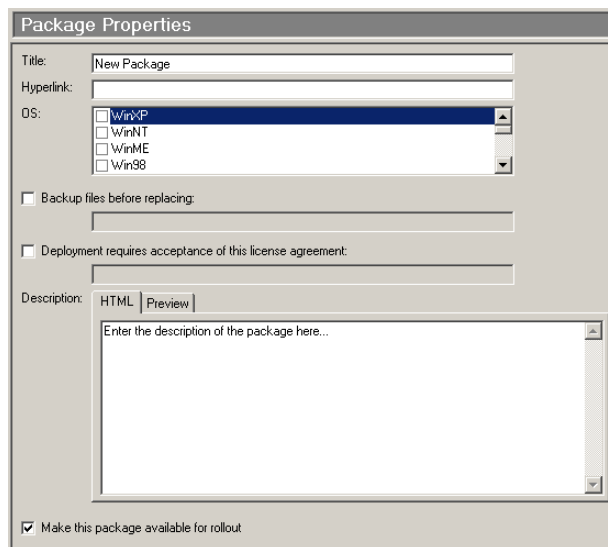


Figure 5.6 Package Properties

3. Define the appropriate package properties
 - a. In the **Title** field, type the title of the package
 - b. In the **Hyperlink** field, type the URL of the package
 - c. In the **OS** list, select the Operating System associated with the package
 - d. If needed, select **Backup files before replacing**, and type the instructions the patch recipient will view.
 - e. If needed, select **Deployment requires acceptance of this license agreement**, and type the text or the URL to be included in the dialog box the recipient will view.
 - f. In the **Description** field, type the text describing the Package. This text is visible within ZENworks Patch Management Server. Click **Preview** to see the text.
 - g. If needed, select **Make this package available for rollout**.
4. Select the **Content** folder to continue.



Adding an Existing Package

To Add an Existing Package

1. Click **Open** in the *Package Summary* window
The *Add Associated Package* window opens
2. Select the package from the *Add Associated Package* window

Removing a Package

To Remove a Package

1. Select the package you want to remove
2. Click **Delete**
3. Click **OK** to remove the package



Editing a Package

To Edit a Package

1. Select the package you want to edit
2. Click **View**

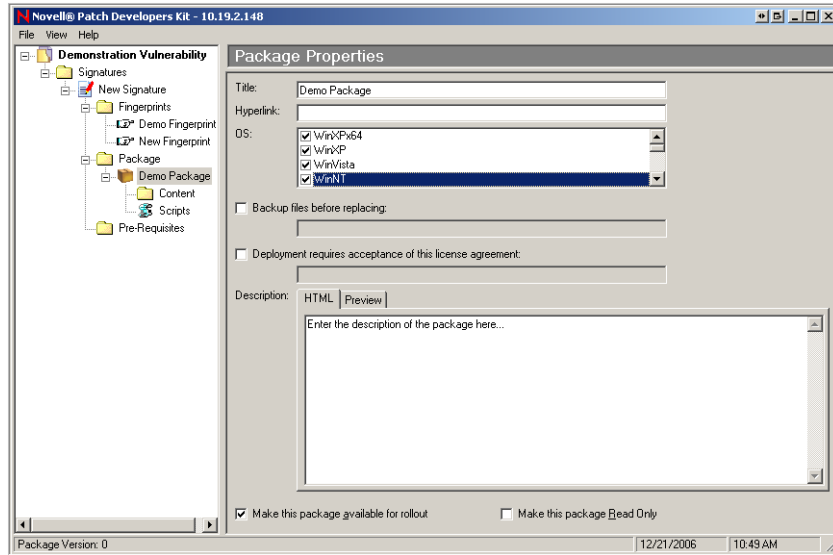


Figure 5.7 Package Properties

3. The *Package Properties* window displays in the window
4. Define the appropriate package properties



Adding Files to a Package

Files and directories can be added to the package by right-clicking the **Package Content** window, and selecting one of the following options:

- “Adding a New Drive to a Package”
- “Adding a New Macro to a Package”
- “Creating a Folder for a Package”
- “Inserting a Folder into a Package”
- “Inserting Files into a Package”
- “Deleting a File from a Package”
- “Renaming a File within a Package”
- “File Properties for a Package”

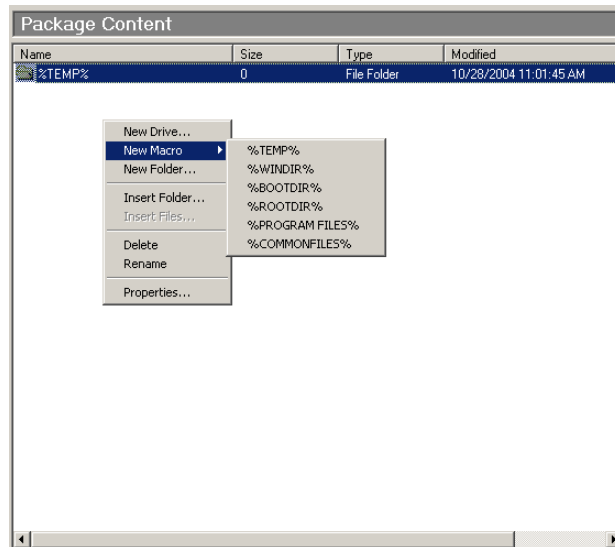


Figure 5.8 Package Content



Adding a New Drive to a Package

Use the *New Drive* option to deploy a package to a drive other than the C : \ or %TEMP% drives.

To Add a New Drive

1. Right-click inside the *Package Content* window
2. Select **New Drive** from the pop-up menu
The *Add Drive* window opens

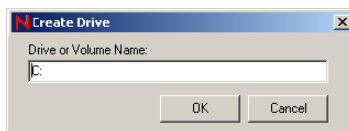


Figure 5.9 Create Drive

3. In the **Drive or Volume Name** field, type the letter you require for the drive name, followed by a colon in X: format
4. Click **OK**
The drive is added to the *Package Content* window

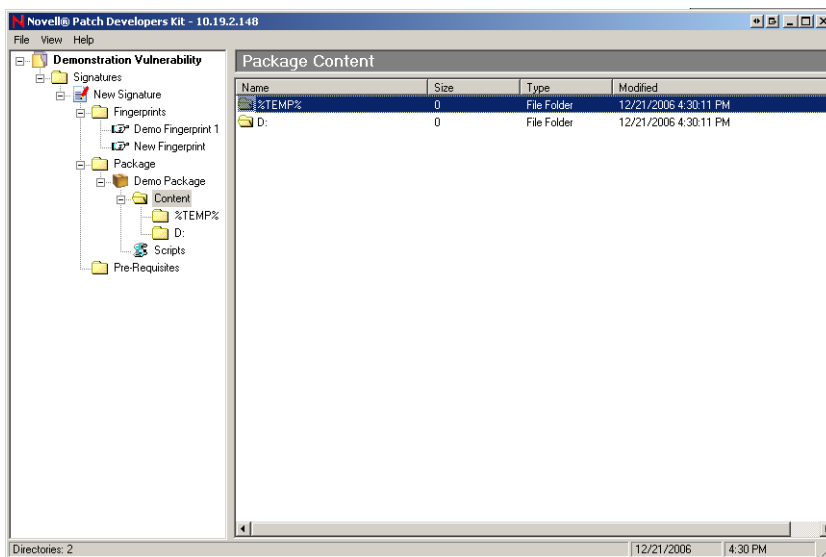


Figure 5.10 Package Content window - New Drive



Adding a New Macro to a Package

Macros access existing system directories. A macro can be either an environment variable, as defined by the operating system, or a macro that only the Novell Agent can expand.

The following pre-defined macros are available under the **New Macro** menu:



Note: Not all macros are available on all Operating Systems. Choose only the macros that are compatible with the operating systems and configurations you are using.

- **%TEMP%** - The operating system temp directory location. Expands to C:\Windows\Temp, C:\Temp, C:\WinNT\Temp, or /tmp depending on operating system and configuration.
- **%WINDIR%** - The operating system windows directory location. %WINDIR% typically expands to C:\Windows
- **%BOOTDIR%** - The operating system boot directory location. Typically expands to C:\
- **%ROOTDIR%** - The operating system root directory location. Typically expands to C:\
- **%PROGRAM FILES%** - The operating system program files location. Typically expands to C:\Program Files
- **%COMMON FILES%** - The operating system common files location. Typically expands to C:\



To Add a Macro

1. Right-click inside the *Package Content* window
2. Select **New Macro** and the macro required for the package

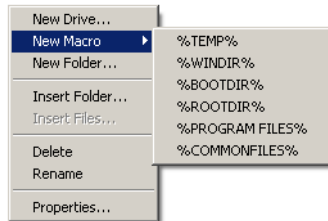


Figure 5.11 Macro Menu

The selected macro is added to the *Package Content* window and the *Package* directory

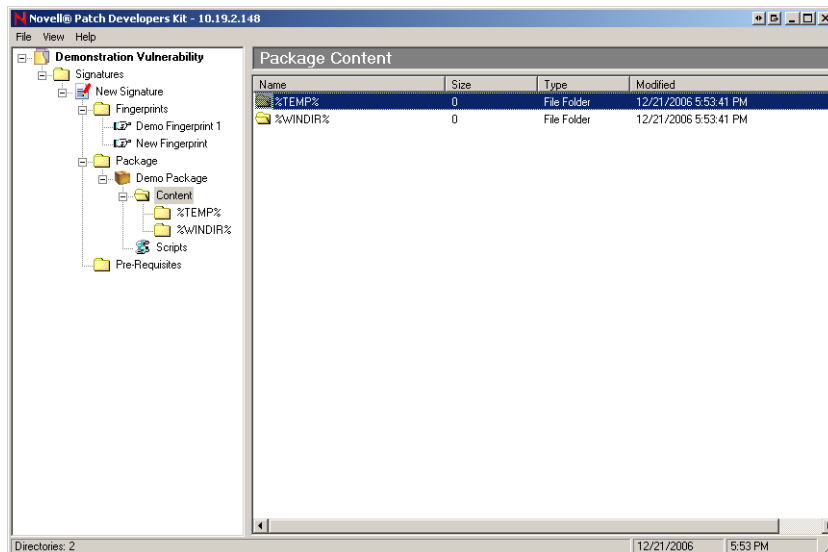


Figure 5.12 Package Content window - Add Macro



Creating a Folder for a Package

The Create Folder window allows for creating a folder within the Package Content directory

To Create a New Folder

1. Right-click inside the *Package Content* window
2. Select **New Folder**
The Create Folder window opens

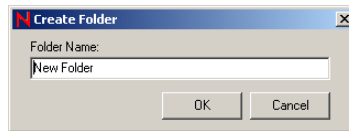


Figure 5.13 Create Folder

3. In the **Folder Name** field, type the name of the new folder
4. Click **OK**
The folder is added to the *Package Content* window and the *Package* directory

Inserting a Folder into a Package

Opens a file system window where you can locate and select an existing directory to add to the Package. To Insert a Folder

1. Right-click inside the *Package Content* window
2. Select **Insert Folder**
The Insert Folder window opens

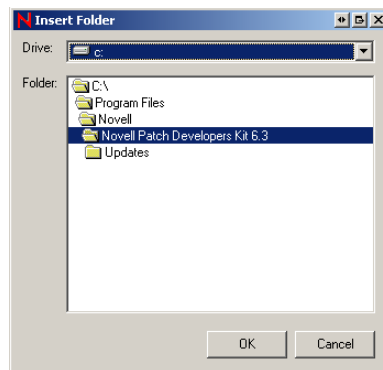


Figure 5.14 Insert Folder



3. Using the directory, locate and select the folder needed for the package
4. Click **OK**
The folder is added to the *Package Content* window and the *Package* directory

Inserting Files into a Package

Opens a file system browser window where you can select existing files to add to the package

To Insert a File into a Package

1. Create or insert a folder for the Package
The folder displays in the *Package* directory tree window.

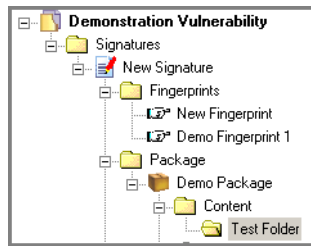


Figure 5.15 Directory Tree

2. In the **directory tree**, select the folder required for the file.
The *Package Content* window opens a blank window.
3. Right-click inside the *Package Content* window



4. Select **Insert File**
The *Open* dialog box opens.

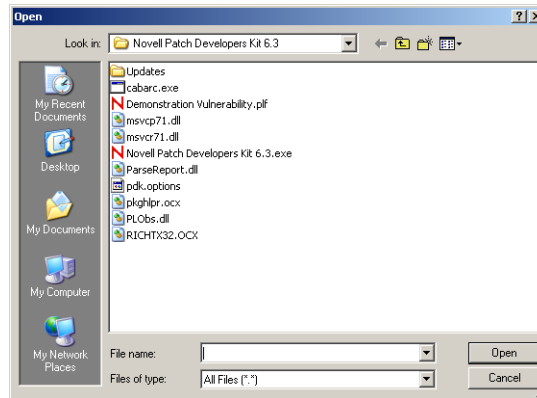


Figure 5.16 Insert Files

5. Select the file needed for the package and click **Open**
The selected file is added to the *Package Content* window.

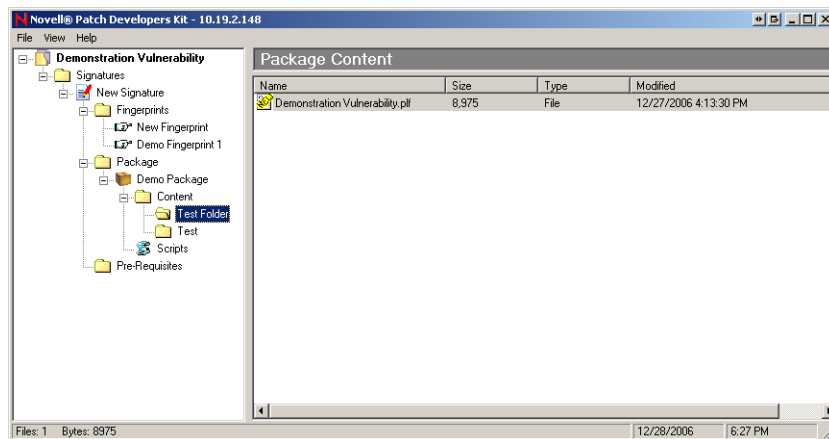


Figure 5.17 Insert File within Package Window



Deleting a File from a Package

Deletes the selected directory or file. This option is available only for files added to the *Package Content* window.

To Delete a Directory or File

1. In the *Package Content* directory tree, select the directory where the file to be deleted is located
The files within the selected directory display in the *Package Content* window
2. Select the file to be deleted
3. Right-click inside the *Package Content* window
4. Select **Delete**
The file is deleted from the package
5. Select the parent directory folder to confirm the deletion

Renaming a File within a Package

The Rename option allows for renaming of a previously created drive or macro within the Package.

To Rename a Directory or File

1. In the *Package Content* directory tree, select the directory where the file is to be renamed
The files within the selected directory display in the *Package Content* window
2. Select the file to be renamed
3. Right-click inside the *Package Content* window
4. Select **Rename**
The *Rename* window opens

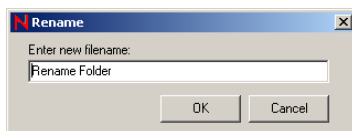


Figure 5.18 Rename

5. In the **Enter new filename** field, type the new name of the file
6. Click **OK**
The folder name is changed and displays in the *Package Content* window and the *Package* directory



File Properties for a Package

Brings up the *properties page* for the selected item. Only available when you right click on a file that has previously been added to the *Package Content* window

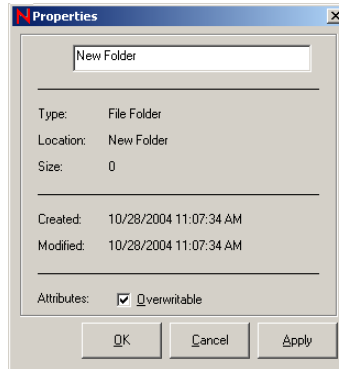


Figure 5.19 Properties

To Change the Overwrite Properties

1. In the *Package Content* directory tree, select the directory where the file is located
The files within the selected directory display in the *Package Content* window
2. Select the file needed
3. Right-click inside the *Package Content* window
4. Select **Properties**
The *Properties* window opens
5. In the **Attribute** field, select or deselect the **Overwritable** checkbox
6. Click **Apply**
The folder properties are changed



Warning: Removing the check-mark from the **Overwritable** attribute will prevent subsequent patches that contain the same file from overwriting that file.



Adding Package Scripts

There are three types of scripts. These scripts can be written in *Microsoft Visual Basic Script* or *Microsoft Jscript*. Documentation regarding these languages can be found at the Microsoft scripting web site: <http://msdn.microsoft.com/scripting>.

The following scripts are listed by the order in which they execute within the PDK:

1. **Pre-Script** - Used to test for a machine condition or shutdown a service. For example you can stop the package rollout in the pre-script by using the `SetReturnCode` in the `PLCCAgent` script object
2. **Command Line Script** - Used to launch executables. The format is the same as a standard `.CMD` or `.BAT` file
3. **Post-Script** - Used for any clean-up operations such as the deletion of files, starting services, or running an installer

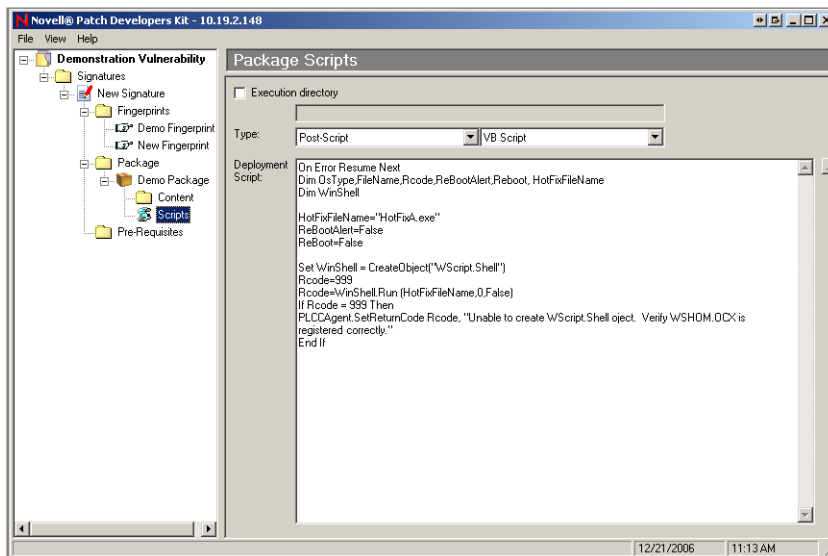


Figure 5.20 Package Script

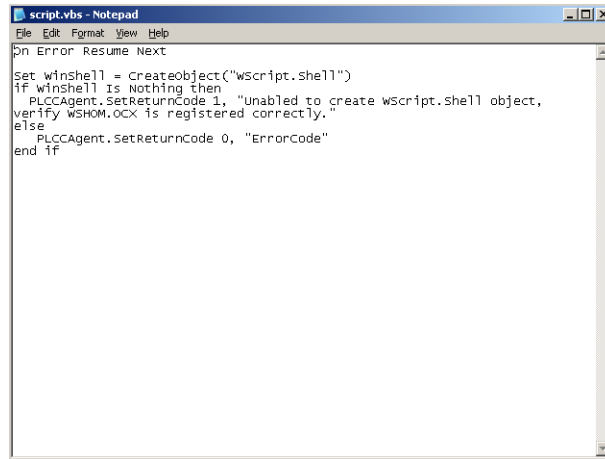
A software package can have a maximum of one of each type of script. When all three scripts are present, they will be executed in the order listed above.



Note: Unless the **Execution Directory** option is selected and a valid directory is defined, all scripts run in the ROOT directory.

Viewing Scripts With Notepad

By clicking the ellipsis (...) button to the right of the Deployment Script you can open and edit the script using *Microsoft® Notepad*



```

script.vbs - Notepad
File Edit Format View Help
On Error Resume Next
Set winshell = CreateObject("wscript.shell")
if winshell is Nothing then
    PLCCAgent.SetReturnCode 1, "Unable to create wscript.shell object,
    verify WSHOM.OCX is registered correctly."
else
    PLCCAgent.SetReturnCode 0, "ErrorCode"
end if
  
```

Figure 5.21 Script Using Notepad

Setting Package Deployment Flags

By adding the following flags to the Package Description, you can specify which flags are available and their default settings when performing a deployment.

Package Flag Descriptions

The following table defines the flag behavior and their descriptions:

Table 5.4 Package Flag Descriptions and Behavior

Description (flag behavior)	Display Flag	Select Flag
Perform an uninstall; can be used with -m or -q	-yd	-y
Force other applications to close at shutdown	-fd	-f
Do not back up files for uninstall	-nd	-n
Do not restart the computer when the installation is done	-zd	-z
Use quiet mode, no user interaction is required	-qd	-q



Table 5.4 Package Flag Descriptions and Behavior

Description (flag behavior)	Display Flag	Select Flag
Use unattended Setup mode	-md	-m
Install in multi-user mode (UNIX, Linux only)	-dmu	-mu
Install in single-user mode (UNIX, Linux only)	-dsu	-su
Restart service after installation (UNIX, Linux only)	-drestart	-restart
Do not restart service after installation (UNIX, Linux only)	-dnorestart	-norestart
Reconfigure after installation (UNIX, Linux only)	-dreconfig	-reconfig
Do not reconfigure after installation (UNIX, Linux only)	-dnoreconfig	-noreconfig
This package is chainable and will run Qchain.exe (windows) or (UNIX/Linux)	-dc	-c
Suppress the final chained reboot	-dc	-sc
Repair permissions	-dr	-r
Deploy Only	-PLD1	-PLD0
No Pop-up	-PLN1	-PLNP
Debug	-PLDG	-PLDEBUG
Suppress Repair	-dsr	-sr
Force the script to reboot when the installation is done	-1d	-1
Reboot is required	Not Applicable	-2
Reboot may occur	Not Applicable	-3
Reboot is required, and MAY occur	Not Applicable	-4



6 Working with Pre-Requisites

In order for a signature to be applicable, one of its pre-requisite signatures must be detected. A pre-requisite as a signature that determines if a patch is applicable to a vulnerability.

In This Chapter

- “Adding a Pre-Requisite” on page 67
- “Adding an Existing Pre-Requisite” on page 69
- “Removing a Pre-Requisite” on page 70
- “Editing a Pre-Requisite” on page 70

Adding a Pre-Requisite

To Create a Pre-Requisite Signature

1. Select the *Pre-Requisites* folder in the Details area.
The *Vulnerability Properties* window opens.
2. Click **Add**
The *Add Pre-Requisite Signature* window opens.
3. Delete **Detect**, and click **Search**.
The signatures created by the user display in the window

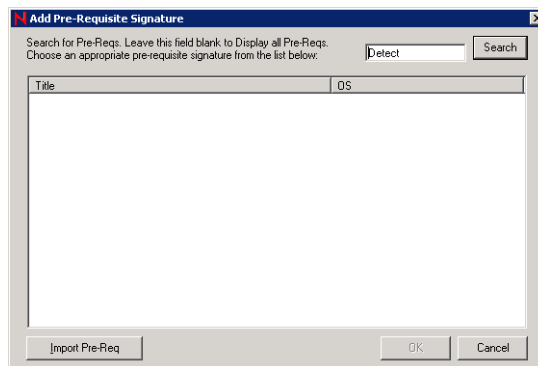


Figure 6.1 Add Pre-Requisite Signature



- If needed, click **Import Pre-Req**
The list of vendor pre-requisites display in the *Add Pre Requisites Signature* window.

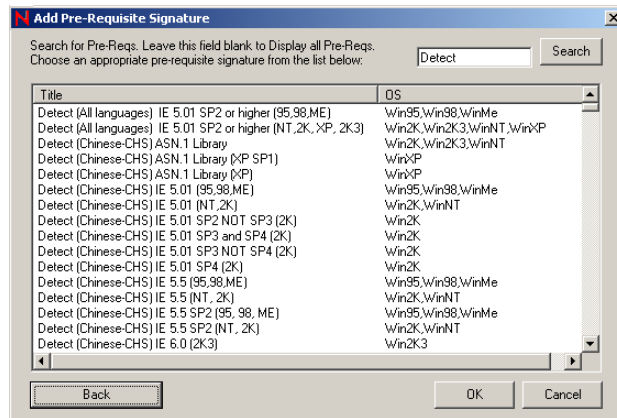


Figure 6.2 Vendor Pre-Requisites

- Select a pre-requisite. Click **OK**.
The pre-requisite is added to the signature.



Adding an Existing Pre-Requisite

To Add a Pre-Requisite Signature

1. Click **Add** at the bottom of the *Pre-Requisite Summary* window

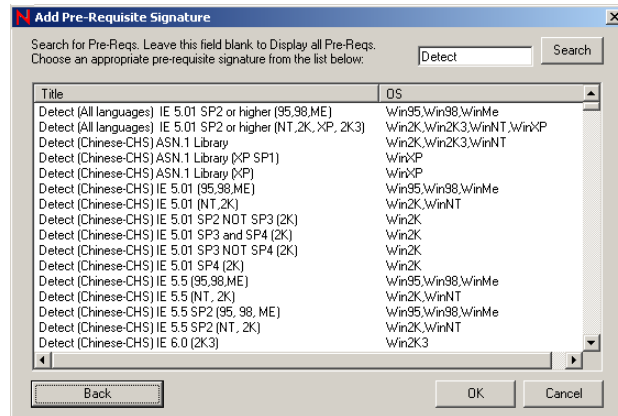


Figure 6.3 Pre-Requisite Signature

2. Select the desired pre-requisite signature
3. Click **OK**
The Pre-Requisite is added.



Tip: When searching for a particular pre-requisite, type the name of the signature in the search field and click **Search**.



Removing a Pre-Requisite

To Remove a Pre-Requisite Signature

1. Select the pre-requisite you want to delete from within the *Pre-Requisite Summary* window
2. Click **Delete**.
The *Delete confirmation* dialog opens
3. Click **OK** to confirm the deletion

Editing a Pre-Requisite

To Edit a Pre-Requisite Signature

1. In the *Vulnerability Summary* window, select the pre-requisite you want to edit
2. Click **Edit**
The selected pre-requisite opens in the *Vulnerability Properties* window.

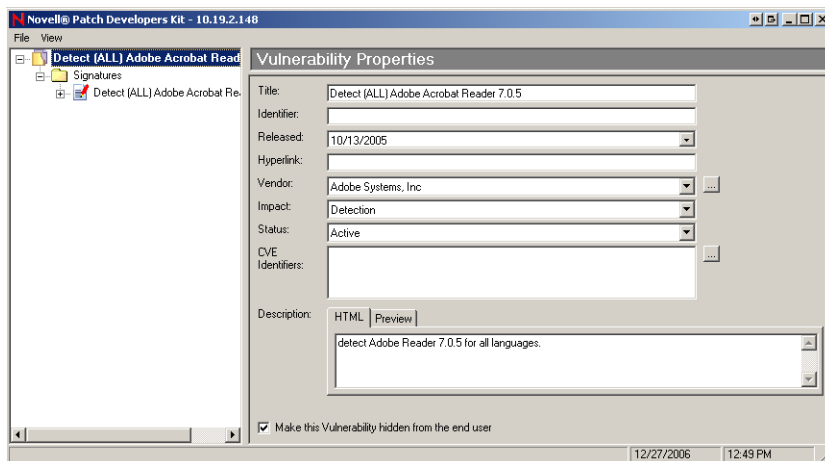


Figure 6.4 Vulnerability Properties

3. In the *Vulnerability Properties* window, edit the Pre-Requisite signatures' properties or define new pre-requisite signatures



7 Importing and Exporting Patches

The PDK allows for transferring custom patches from one ZENworks Patch Management Server to another. The import and export features allow for exporting a patch to a physical PLF file. After the export is completed, the patch can be stored, emailed, or uploaded to another ZENworks Patch Management Server.

In this Chapter

- “Defining the Import Summary Window” on page 71
- “Importing Patches” on page 75
- “Using Command Line Importing” on page 78
- “Exporting Patches” on page 79

Defining the Import Summary Window

The following section describes the fields, menus, and icons associated with the *Import Summary* window.

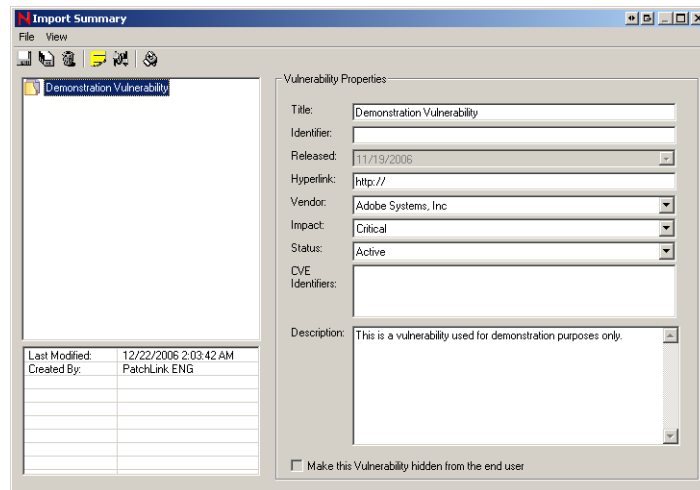








Figure 7.1 Import Summary window



Import Summary Toolbar

The following table describes the Import Summary icons and functions.

Table 7.1 Import Summary Icons

Icon	Name	Description
	Save	Saves the selected patch to your Patch Management Server
	Save All	Saves all of the imported patches to your Patch Management Server
	Delete	Removes the selected patch from the import (aborting the import for the selected patch only)
	View Log	Opens the Import Log window.
	View Dependencies	Displays all of the currently selected patch's dependencies
	Print View	Opens the selected patch's properties in <i>notepad.exe</i> (includes the report properties, signatures, fingerprints, and packages)

Defining the Menu Options

The following table describes the Import Summary window menu options and functions.

Table 7.2 Menu Items

Menu Item	Command	Description
Print View	File>Print View	Opens the selected patch's properties in <i>notepad.exe</i> (includes the report properties, signatures, fingerprints, and packages)
Exit	File>Exit	Ends the PDK session
Toolbar	View>Toolbar	Toggles the display of the toolbar on/off
Log Window	View>Log Window	Displays the Import Log pane of the <i>Import Summary</i> window



Defining the Import Summary fields

The following table describes the import summary fields and descriptions.

Table 7.3 Vulnerability Properties

Field	Description
Title	Contains the name of the vulnerability. The vulnerability requires a title to display properly in your ZENworks Patch Management Server
Identifier	Contains the vendor specific number or id value that uniquely correlates with this vulnerability
Released	Contains the date that the vendor released the patch. When creating a new vulnerability, this field is set to the current date by default. This date should be changed to correspond with the date the patch was released by the vendor
Hyperlink	An optional field that provides a link to more information. If a URL is entered in this field, the More Information link in the <i>Vulnerabilities</i> page within Patch Management Server is visible.
Vendor	Represents the company that released the patch. Clicking the drop-down arrow will allow you to select from a list of vendors that are already in the database



Table 7.3 Vulnerability Properties

Field	Description
Impact	<p>Indicates the level of severity for this patch. The values are as follows:</p> <ul style="list-style-type: none"> • Critical - Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. Most of the recent security updates fall in to this category. The patches for this category are automatically downloaded and stored on your ZENworks Patch Management Server. • Critical - 01 - Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. This patch is older than 30 days and has not been superseded. • Critical - 05 - Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. These patches have been superseded. • Critical - Intl - An international patch, where Novell or the product manufacturer has determined that this patch is critical and should be installed as soon as possible. Most of the recent international security updates fall in to this category. After 30 days international patches in this category will be moved to Critical - 01. • Detection - These vulnerabilities contain signatures that are common to multiple vulnerabilities. They contain no associated patches and are only used in the detection process. • Informational - These vulnerabilities detect a condition that Novell or the product manufacturer has determined as informational. If the report has an associated package, you may want to install it at your discretion. • Recommended - Novell or the product manufacturer has determined that this patch, while not critical or security related is useful and should be applied to maintain the health of your computers. • Software - These vulnerabilities are software applications. Typically, this includes software installers. The vulnerabilities will show not patched if the application has not been installed on a machine. • Task - This category contains tasks which administrators may use to run various detection or deployment tasks across their network. • Virus Removal - This category contains packages which administrators may use to run various virus detections across their network. Anti-Virus tools and updates are included in this category. •
Status	<p>Defines the status of the vulnerability. If set to Active, Patch Management Server users will be able to view this vulnerability in the <i>Vulnerabilities</i> page. If set to Beta, only Patch Management Server sites set for Beta use will be able to view the vulnerability</p>
CVE Identifier	<p>Allows for the patch to be defined and classified using the Common Vulnerabilities and Exposures standard. See http://cve.mitre.org for more information.</p>
Description	<p>Contains a text description of the vulnerability. The information is displayed in the <i>Vulnerabilities</i> page of ZENworks Patch Management Server also.</p>



Importing Patches

Using the import wizard, you can import patches into ZENworks Patch Management Server.

To Import Vulnerabilities

1. From the **File** menu, select the **Import Wizard**.
The *Import Wizard* screen opens.

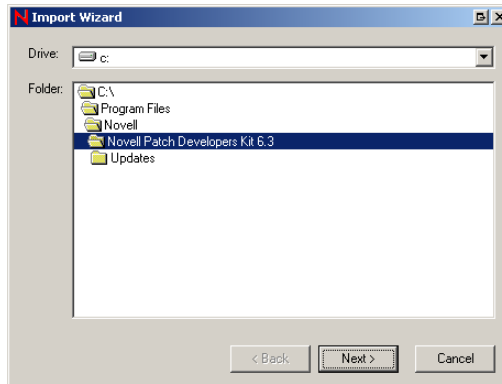


Figure 7.2 Import Wizard screen

2. Navigate to the folder where the patches are located and click **Next**.
The *Import Wizard* displays the patches from that folder in the *Patch Selection* window.

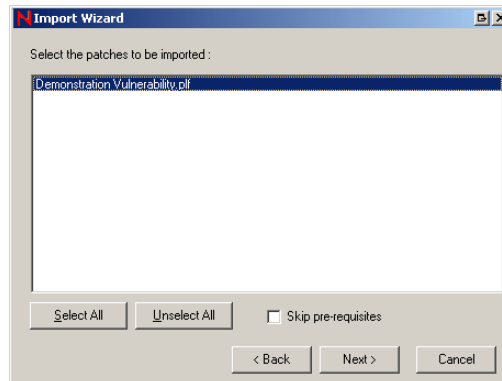


Figure 7.3 Patch Selection screen



3. Select the patch files that you wish to import and click **Next**.



Note: If there are multiple variations of the same vulnerability it may be due to different signatures for different operating systems, languages, and regions.

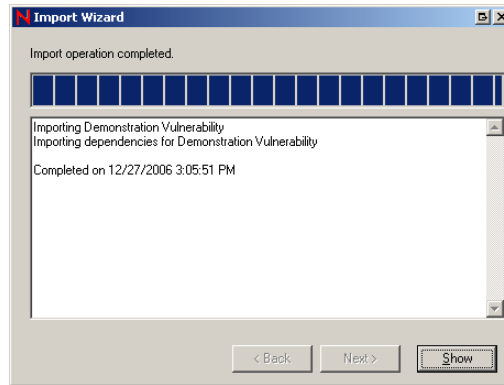


Figure 7.4 Importing screen



Note: The **Import Wizard** may take several minutes; depending upon the number of patches, size of each patch, and the total number of files and directories it contains.



4. Click **Show**.

The *Import Wizard* closes and the *Import Summary* window opens. The list of imported patches displays in the *Vulnerability List*.

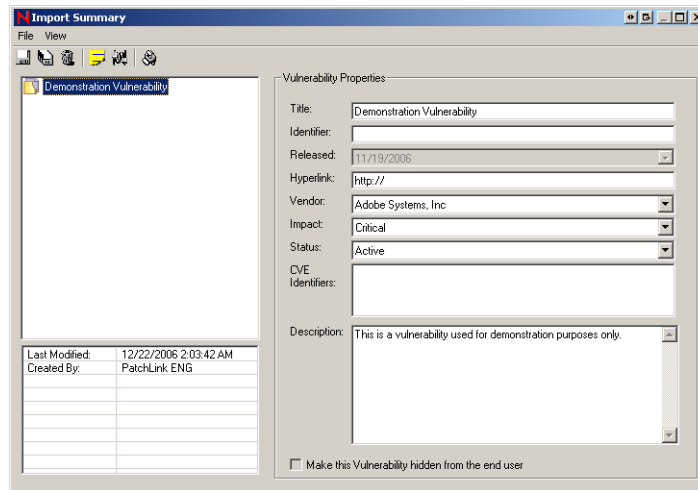


Figure 7.5 Import Summary

5. Select the patch to import to Patch Management Server.
The *Vulnerability Properties* area populates with the patch details.
6. Click **Save**.
The selected patch is saved to the Patch Management Server.
7. To import all patches, click **Save All**.
8. Select **File > Exit**, or click **Close**.
The PDK ends the Import session and closes the *Import Summary* window.



Using Command Line Importing

Patches can be imported from the Command Line.

To Use Command Line Importing

1. Open a command prompt and navigate to the PDK program files directory
2. Type in the name of the PDK executable as defined:

Syntax

```
PDK.exe -i -u -s "SERVERNAME" -p "SERIAL NUMBER or SA PASSWORD" -plf "FULL PATH TO PLF FILE"  
-l ["FULL PATH TO LOG FILE"]
```

Switch

Table 7.4 Command Line Switch Descriptions

Switch	Description
-i	Use Import Mode (REQUIRED)
-u	Use Unattended Mode (REQUIRED)
-s	Set the Server Name (REQUIRED)
-p	Set the Serial Number or SA password if the default SA password has been changed (REQUIRED)
-plf	Set the full path to the PLF file. may be separated by a " " to include multiple patches (REQUIRED)
-l	Create a log file - If a path is NOT specified the log will be created in the PDK Program Files Directory (OPTIONAL)

Example

Import the patch MS02-04.PLF without a log file

```
PDK.exe -i -u -s "myserver" -p "xxxxxxxx-xxxxxxx" -plf "C:\Patches\MS02-04.PLF"
```

Import the patch MS02-04.PLF with a log file

```
PDK.exe -i -u -s "myserver" -p "xxxxxxxx-xxxxxxx" -plf "C:\Patches\MS02-04.PLF" -l  
"log.txt"
```



Exporting Patches

The *Export Wizard* exports custom patches from ZENworks Patch Management Server to a specified folder. By exporting patches, you can import them into other ZENworks Patch Management Server without having to recreate the vulnerability.



Note: The selected archive folder must have sufficient free space for the exported patches. Your %TEMP% folder must also have several hundred megabytes free to successfully export large patches.

To Export Patches

1. From the **File** menu, select **Export Wizard**
The *Export Wizard* window opens and the available patches display in the window

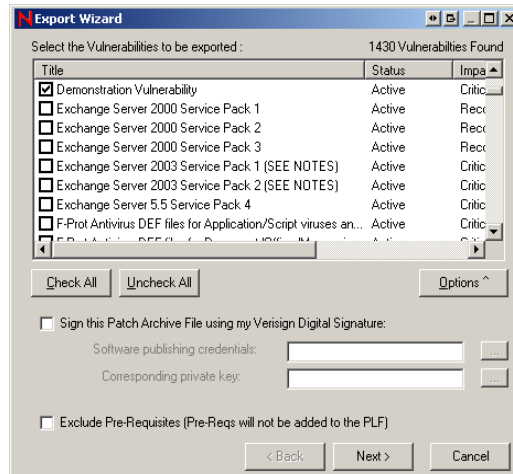


Figure 7.6 Export Wizard page



- If filtering is needed, click **Options**
The **Filter Options** area displays in the *Export Wizard* page

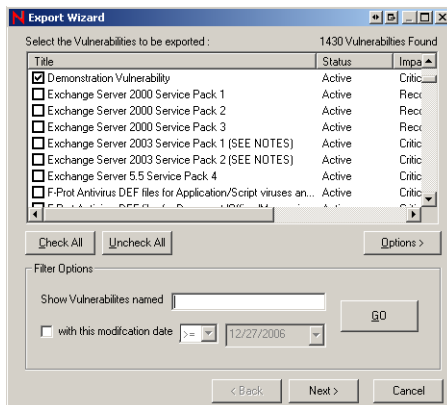


Figure 7.7 Select Vulnerabilities

- Type the name of the vulnerability in the **Show Vulnerabilities named** field
 - If needed, select **with this modification date**, select \geq or \leq , and the date from the drop down list boxes
 - Click **GO**
The sorted list displays in the window
 - Click **Options>** to close the *Filter Options* area
- Select the patches to be exported



4. If needed, select **Sign this Patch Archive File using my Verisign Digital Signature**
The **Software Publishing Credentials** and the **Corresponding Private Key** fields become active

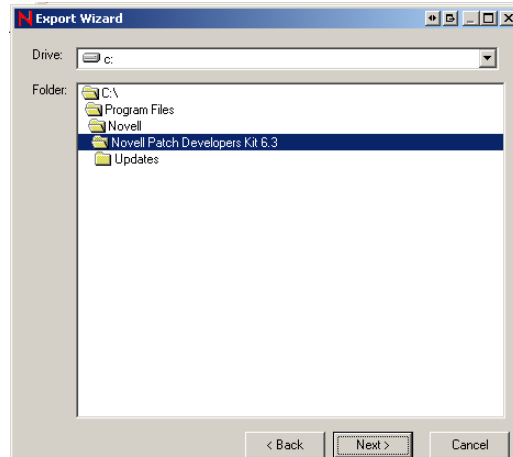


Figure 7.8 Locate Software Publishing Credentials

- a. In the **Software publishing credentials** field, click **Search**
The *Locate Software Publishing Credentials* page opens
- b. Locate the Verisign certificate (.spc) within the directory and click **Open**
The **Software Publishing Credentials** field displays the .spc file



5. If needed, in the **Corresponding private key** field, select **Search**
The *Locate Private Key* page opens

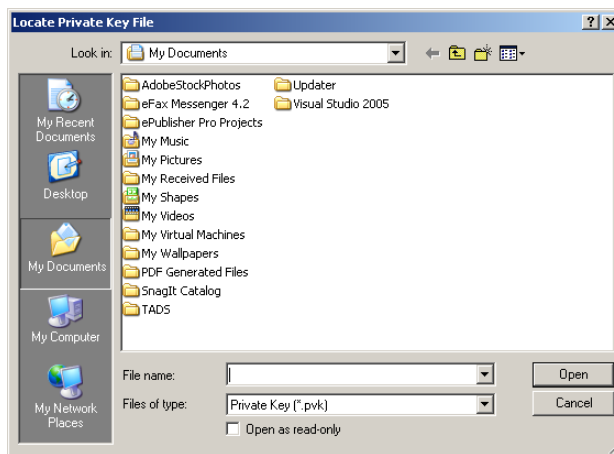


Figure 7.9 Locate Private Key

- a. Locate the Verisign private key (.pvk) within the directory
 - b. Click **Open**
The *Corresponding private key* field displays the .pvk file
6. If needed, select **Exclude Pre-Requisites (Pre-Reqs will not be added to the PLF)**
Selecting this option prevents the PDK from automatically including the patch pre-requisites
 7. Click **Next**
The *Export Wizard* opens a directory for saving the exported patches



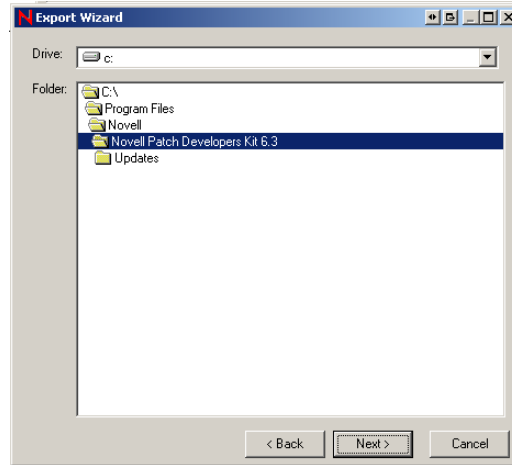


Figure 7.10 Patch export location

8. Select the desired archive (output) folder
9. Click **Next**
The *Export Wizard* exports the patch and saves it to the selected archive folder

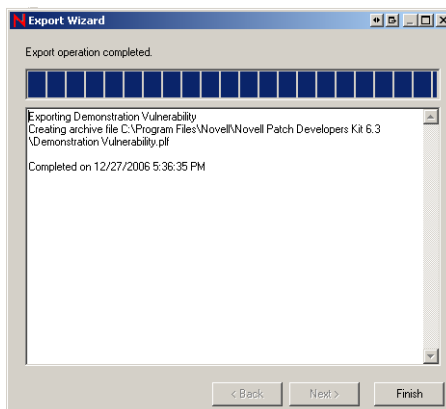


Figure 7.11 Export complete

10. Click **Finish** to close the *Export Wizard*





A Reference: PLCCAgent Object Methods

The agent scripting host contains the embedded Novell Agent Control Object (PLCCAgent). This object provides functions to access the windows registry, agent environment, and script output.

Defining the PLCCAgent Object Methods

The methods available for use with the PLCCAgent are defined as follows:

GetOSVersion

Description

Obtains information about the current operating system version

Syntax

```
object.GetOSVersion (strOS, iMajor, iMinor, iBuild,
strServicePack)
```

Parameters

Table A.1

Parameter	Description
object	PLCCAgent object
strOS	Win95, Win98, WinME, WinNT, Win2K, WinXP
iMajor	Major version (NT 4.0 Major = 4)
iMinor	Minor version
iBuild	Build Number
strServicePack	Service Pack Number

Return

Returns a non-zero value if successful

Remarks

The OS version number can be cryptic, dependent upon the current operating system. For example, Windows 2000, Windows XP, and Windows Server 2003, all return a major version of 5, but with different minor versions.



Example

```
PLCCAgent.GetOSVersion (strOS, iMajor, iMinor, iBuild,  
strSP)
```

GetPolicy

Description

Obtains the value for an agent policy

Syntax

```
object.GetPolicy (strName, strValue)
```

Parameters

Table A.2

Parameter	Description
object	PLCCAgent object
strName	"Interval", "IntervalType", "TraceLevel"
strValue	Returned value of a policy

Return

Returns a non-zero value if successful

Remarks

Can be used to determine the current agent communication policy in the package script

Example

```
PLCCAgent.GetPolicy ("Interval", strValue)
```



InitiateSystemShutdown

Description

Used to restart the agent computer

Syntax

```
object.InitiateSystemShutdown
```

Parameters

Table A.3

Parameter	Description
object	PLCCAgent object

Return

Returns a non-zero value if successful

Remarks

Causes the machine to restart

See Windows SDK API `ExitWindowsEx (EWX_REBOOT, 0)`

Example

```
PLCCAgent.InitiateSystemShutdown ()
```



PollHost

Description

Tells the agent to poll the ZENworks Patch Management Server as soon as this package containing this script completes

Syntax

```
object.PollHost ()
```

Parameters

Table A.4

Parameter	Description
object	PLCCAgent object

Return

Returns a non-zero value if successful

Remarks

Used to determine if the agent still has a connection to the Patch Management Server

Example

```
PLCCAgent.PollHost ()
```



RegCloseKey

Description

Closes the handle to the registry key opened by RegOpenKey

Syntax

```
object.RegCloseKey (hKey)
```

Parameters

Table A.5

Parameter	Description
object	PLCCAgent object
hKey	Handle to open key

Return

Returns a non-zero value if successful

Remarks

To avoid memory issues and other scripting errors you should close all registry keys that you have opened, prior to exiting your script.

Example

```
If PLCCAgent.RegOpenKey(0,
"HKLM\Software\Microsoft\Windows\CurrentVersion", hKey)
then
    'Key opened successfully
    PLCCAgent.RegCloseKey(hKey)
End if
```



RegEnumKey

Description

Lists subkeys of the specified registry key. The function retrieves the name of a subkey each time it is called

Syntax

```
object.RegEnumKey (hKey, strEnumKey, iIndex)
```

Parameters

Table A.6

Parameter	Description
object	PLCCAgent object
hKey	Handle to an open registry key
strEnumKey	A variable that receives the name of the subkey in string form Note: This function copies only the name of the subkey, not the full key hierarchy
iIndex	Specifies the index of the subkey to retrieve

Return

Returns a non-zero value if successful

Remarks

To enumerate subkeys:

1. Call RegEnumKey with iIndex equal to zero (0)
2. Increment iIndex
3. Repeat steps 1 and 2 until the function returns zero (0)



Note: While an application is using the RegEnumKey function it should not make calls to any registry functions that might change the key being queried

Example

```

If PLCCAgent.RegOpenKey(0,
"HKLM\Software\Microsoft\Windows\CurrentVersion", hKey)
then

    iKeyIndex = 0 'Must start with 0
    do while PLCCAgent.RegEnumKey(hKey, szKey, iKeyIndex)
        PLCCAgent.Write "Key = "& szKey & vbCrLf
        iKeyIndex = iKeyIndex + 1 'Next Key
    loop
    PLCCAgent.CloseKey(hKey)
End If

```

RegEnumValue

Description

Enumerates the values for the specified open registry key by copying one indexed value name and data block each time it is called

Syntax

```
object.RegEnumValue(hKey, strEnumValue, iIndex)
```

Parameters

Table A.7

Parameter	Description
object	PLCCAgent object
hKey	Handle to an open registry key
strEnumValue	A variable that receives the value name in string form
iIndex	Specifies the index of the value to retrieve

Return

Returns a non-zero value if successful



Remarks

To enumerate values:

1. Call `RegEnumValue` with `iIndex` equal to zero (0)
2. Increment `iIndex`
3. Repeat steps 1 and 2 until the function returns zero (0)

Example

```
'Read all Values from a Key and output them to the Host
If PLCCAgent.RegOpenKey(0,
"HKLM\Software\Microsoft\Windows\CurrentVersion", hKey)
then
    iKeyValue = 0 'Must start with 0
    do while PLCCAgent.RegEnumValue(hKey, szValue,
iValueIndex)
        PLCCAgent.Write "Value = "& szValue &vbCrLf;
        iKeyValue = iKeyValue + 1 'Next Value
    loop
    PLCCAgent.CloseKey(hKey)
End if
```



RegOpenKey

Description

Returns the registry value named by *strKey*

Syntax

```
object.RegOpenKey (hRootKey, strKey, hKey)
```

Parameters

Table A.8

Parameter	Description
object	PLCCAgent object
hRootKey	Handle to previous open key (0=none)
strKey	Key name to open.
hKey	Return Handle to open key

Return

Returns a non-zero value if successful

Remarks

If *hRootKey* does not equal zero (0), *strKey* must be a subkey of *hRootKey*, otherwise:

The *strKey* variable must begin with one of the following root key names:

Table A.9

Key Abbreviation	Key Name
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKCR	HKEY_CLASSES_ROOT
HKU	HKEY_USERS
HKCC	HKEY_CURRENT_CONFIG





Note: The `RegOpenKey` function uses the default security access mask when opening a key

Example

```
If PLCCAgent.RegOpenKey (0,  
"HKLM\Software\Microsoft\Windows\CurrentVersion", hKey)  
then  
    'Key opened successfully  
    PLCCAgent.RegCloseKey (hKey)  
End if
```

RegQueryValue

Description

Retrieves the data type and data value of a specified registry key value name

Syntax

```
object.RegQueryValue (hKey, strKey, vValue, iType)
```

Parameters

Table A.10

Parameter	Description
object	PLCCAgent object
hKey	Handle to an open registry key
strKey	A string containing the name of the value to query
vValue	A variant variable that will receive the registry value
iType	An integer variable that receives a code, indicating the data type is stored in the specified registry key

Return

Returns a non-zero value if successful



Remarks

If `hKey` does not equal zero (0) then `strKey` must be a subkey of `hKey`, otherwise the `strKey` variable must begin with one of the following root key names:

Table A.11

Key Abbreviation	Key Name
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKCR	HKEY_CLASSES_ROOT
HKU	HKEY_USERS
HKCC	HKEY_CURRENT_CONFIG

Dependent upon the data type of the registry key, the `iType` variable must equal one of the following:

Table A.12

Data Type	iType	Data Type Description
REG_BINARY	3	Binary data in any format
REG_DWORD	4	A 32-bit number
REG_EXPAND_SZ	2	A null-terminated Unicode string that contains un-expanded references to environment variables (Example: "%PATH%")
REG_SZ	1	A null-terminated Unicode string



Example

```
If PLCCAgent.RegOpenKey(0,
"HKLM\Software\Microsoft\CurrentVersion", hKey) then
    if PLCCAgent.RegQueryValue(hKey, "ProductId", Value,
iType) then
        PLCCAgent.Write "Value is = "& Value & "type "&
iType & vbcrLf
    End if
    PLCCAgent.CloseKey(hKey)
End if
```

RegRead

Description

Returns the registry value named by strName

Syntax

```
object.RegRead (strName, strValue, iType)
```

Parameters

Table A.13

Parameter	Description
object	PLCCAgent object
strName	The registry value name to read
strValue	Data read from registry
iType	An integer variable that receives a code indicating the data type is stored in the specified registry key

Return

Returns a non-zero value if successful



Remarks

Dependent upon the data type of the registry key, the *iType* variable must equal one of the following:

Table A.14

Data Type	iType	Data Type Description
REG_BINARY	3	Binary data in any format
REG_DWORD	4	A 32-bit number
REG_EXPAND_SZ	2	A null-terminated Unicode string that contains un-expanded references to environment variables (Example: "%PATH%")
REG_SZ	1	A null-terminated Unicode string



Note: The **RegRead** method supports only REG_SZ, REG_EXPAND_SZ, REG_DWORD and REG_BINARY data types. If the registry is not one of the supported data types **RegRead** returns a zero (0)

Example

```
Dim Value
if (PLCCAgent.RegRead
("HKLM\Software\Microsoft\Windows\CurrentVersion\ProductId
", Value, Type) then
    PLCCAgent.Write "The Product is "& Value
endif
```



RegSetValue

Description

Sets the data and data type of a specified value under a registry key

Syntax

```
object.RegSetValue (hKey, strSubKey, vValue, iType)
```

Parameters

Table A.15

Parameter	Description
object	PLCCAgent object
hKey	A handle to an open registry key
strSubKey	A string containing the name of the value to set
vValue	A variant variable that contains the registry value
iType	An integer variable that contains a code indicating the data type is to be stored in the registry key

Return

Returns non zero value if successful

Remarks

Dependent upon the data type of the registry key, the *iType* variable must equal one of the following:

Table A.16

Data Type	iType =	Data Type Description
REG_BINARY		Binary data in any format
REG_DWORD	4	A 32-bit number
REG_EXPAND_SZ	2	A null-terminated Unicode string that contains un-expanded references to environment variables (Example: "%PATH%")
REG_NONE		No defined type
REG_SZ	1	A null-terminated Unicode string



Example

```
If PLCCAgent.RegOpenKey(0,
"HKLM\Software\Microsoft\Windows\CurrentVersion", hKey)
then
    Value = "This is a string"
    Type = 1 ' 1 = String or REG_SZ
    PLCCAgent.RegSetValue(hKey, "Test", Value, Type)
    PLCCAgent.CloseKey(hKey)
End if
```



SetReturnCode

Description

Used to alter the package status



Note: If during `PreScript` or `PostScript` execution the script determines that the package (or script) has not executed properly `SetReturnCode` will return an error code and description.

Syntax

```
object.SetReturnCode (iRc, strRCDescription)
```

Parameters

Table A.17

Parameter	Description
object	PLCCAgent object
iRc	The returned error code, a value between 1 and 255 must be selected
strRCDescription	The returned error description

Return

Returns both an error code and description

Remarks

If a value is returned during the pre-script an error will be returned to the host, and package files will not be downloaded

Example

```
PLCCAgent.SetReturnCode 1, "The Package Download was aborted"
```



Write

Description

Used to return data to the ZENworks Patch Management Server

Syntax

```
object.Write (output)
```

Parameters

Table A.18

Parameter	Description
object	PLCCAgent object
output	String data to be sent to ZENworks Patch Management Server

Return

Returns a string to the ZENworks Patch Management Server

Remarks

Regardless of the data input to the `Write` function it will be converted to a string prior to transmission

Since the agent supports a `Pre-Script` and `Post-Script` for each package there can be a maximum of two (2) output streams returned to the ZENworks Patch Management Server per package



Note: There is currently no user interface on the ZENworks Patch Management Server to view the data. Future versions of ZENworks Patch Management may include an interface to access this data.

Example

```
PLCCAgent.Write "Hello World"
```





Index

E

exiting the PDK 10
 export
 see export wizard
 export wizard 71, 79

F

fingerprint 2
 properties window 32
 summary window 27
 fingerprint type
 expression 37
 patch 41
 registry 35
 script 42
 WMI 37
 fingerprints
 removing 30
 testing 35

I

impact
 severity levels 14, 74
 import
 command line 78
 see import wizard
 import summary window 71, 77
 import wizard 71, 75
 Importing 71

P

package 3
 adding 51
 adding existing 53
 content window 49, 55
 macros 57
 %BOOTDIR% macro 57
 %COMMON FILES% macro 57
 %PROGRAM FILES% macro 57
 %ROOTDIR% macro 57
 %TEMP% macro 57
 %WINDIR% macro 57
 properties window 47, 52
 scripts
 command line script 64
 post-script 64
 pre-script 64
 scripts window 49
 package deployment flag 65
 packages
 adding files 55
 editing 54
 removing 53
 patch structure 2
 PDK
 exiting 10
 installing 3, 6
 prerequisites 3
 starting 9
 PDK database 3
 installing 3



PLCCAgent	
GetOSVersion.....	85
GetPolicy	86
InitiateSystemShutdown.....	87
PollHost.....	88
ReEnumValue.....	91
RegCloseKey	89
RegEnumKey	90
RegOpenKey	93
RegQueryValue.....	94
RegRead.....	96
RegSetValue	98
SetReturnCode	100
Write	101
PLCCAgent Object Methods	85
pre-requisite signature	3
pre-requisites	
adding.....	67
adding existing.....	69
editing	70
removing	70

S

scripts	
viewing with notepad	65
signature.....	2
properties window	20
summary window.....	19
signatures	21
adding.....	21
editing	23
removing	23
starting the PDK	9

V

vendors	16
adding.....	16
deleting	18
editing.....	17
vulnerability	2
create properties	15
creating.....	15
Vulnerability properties.....	13
vulnerability properties window.....	12





Novell, Inc.
1800 South Novell Place
Provo, UT 84606

www.novell.com
phone: 800.858.4000

