



Altiris® Streaming Composer 5.2.2 Help

Notice

Altiris® Streaming Composer Help

© 2006-2008 Altiris, Inc. All rights reserved.

Document Date: January 10, 2008

Information in this document: (i) is provided for informational purposes only with respect to products of Altiris or its subsidiaries ("Products"), (ii) represents Altiris' views as of the date of publication of this document, (iii) is subject to change without notice (for the latest documentation, visit our Web site at www.altiris.com/Support), and (iv) should not be construed as any commitment by Altiris. Except as provided in Altiris' license agreement governing its Products, ALTIRIS ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTIES RELATING TO THE USE OF ANY PRODUCTS, INCLUDING WITHOUT LIMITATION, WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS. Altiris assumes no responsibility for any errors or omissions contained in this document, and Altiris specifically disclaims any and all liabilities and/or obligations for any claims, suits or damages arising in connection with the use of, reliance upon, or dissemination of this document, and/or the information contained herein.

Altiris may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the Products referenced herein. The furnishing of this document and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any foregoing intellectual property rights.

No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Altiris, Inc.

Customers are solely responsible for assessing the suitability of the Products for use in particular applications or environments. Products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

*All other names or marks may be claimed as trademarks of their respective companies.

Contents

Introduction

- Assumptions 1
- Typographic Conventions 1
- In this Guide 1

Streaming Composer Overview 3

- System Requirements 3
 - Hardware 4
 - Software 4
 - Prerequisites 4
- Installing the Streaming Composer 4
- Summary 5

Using the Streaming Composer Interface 7

- Using the Menu Bar 7
 - File Menu 7
 - Edit Menu 8
 - Tools Menu 8
- Using the Toolbar 9
- Using the Navigation Pane 9
 - Using the Primary Pane 10
- Using the Status Pane 10
- Summary 10

Process Walkthrough 11

- Creating a Package 11
- Reviewing Package Content 11
- Updating Package Information 12
- Building a Package 12
- Getting Started 12
 - Using a Repository 12
 - Using the Streaming Composer Window 13
 - Conventional Package Installation 14
 - Converting Older Packages 14
- Summary 15

Customizing a Package 17

- Using Configuration Profiles 17
 - Set Global Configuration Profile 17
- Defining Packaging Options 19

Using the Monitoring Group 19
File Handling 19
Registry Handling 20
INI File Handling 21
MSI Handling 22
Build Processing 22
Defining the Preferences 23
Summary 24

Creating a Snapshot Package 25

The Create Phase 25
The Package Content Phase 28
 Manifest Information 28
 Review File, Registry, and INI Files 29
 Environment Variables 30
 Package Environment 31
The Package Information Phase 31
 Define Properties 32
 Specify System Requirements 32
 Define Execution Scripts 33
 Update the Application List 35
 Generate Streamlets 37
 Collect the Package Startup Block 37
Building the Package 38
 Resolve Orphan Paths 39
Best Practices for a Snapshot Package 39
Summary 41

Creating an MSI Package 43

Creating a New MSI Package 43
 Defining the MSI Setup 44
Reviewing the Content for an MSI Package 45
Reviewing Information for an MSI Package 46
 Update (Define) Properties for an MSI Package 46
 Review MSI Properties 47
 Set Up System Requirements for an MSI Package 47
 Define Execution Scripts for an MSI Package 47
 Set Up Configurations for an MSI Package 47
 Generate Streamlets for an MSI Package 48
Building the Package 49
Install a Specific Application 49
Best Practices for an MSI Package 50
Summary 50

Creating a Tracking Package 51

Create Phase 51

Applications	52
Application Identification	52
System Requirement	53
Define Execution Scripts	53
Package Building Phase	54
Summary	54

Creating a Package for SVS Applications using a .VSA file 55

The Create Phase	55
The Package Information Phase	56
Building the Package	56
Upgrading SVS Package	56
Summary	57

Upgrading a Package 59

Machine Independent Packaging	59
Upgrading a Snapshot Package	60
Conventional Package Installation	61
Upgrading an MSI Package	61
Creating an MSI Patch Package	62
Summary	64

Using Command-Line for Packaging 65

Information on files and paths	65
Source files path content	66
XML File	67
ShortCuts	68
Additional Customization	68
Samples	69
Sample File Structure	69
Source File Path for Package Upgrade	69
Streaming System environment variables mapping list	70
List of supported properties	71
Create New Snapshot Package	72
Command Line Syntax	72
Procedure	72
Upgrade Snapshot Package	73
Method 1-Upgrade package	73
Method 2-Upgrade package	74
Creating MSI Packages	75
Upgrade MSI Package	76
Creating SVS Packages	77
Summary	78

Appendix A : Execution Script-An Example 79

Appendix B: AppstreamCfg File 83

Appendix C : Packager.INI File 85

Appendix D : Appstream.INI File 89

Appendix E : How to Deploy Service Packs & Hotfixes 91

Glossary 97

Introduction

This guide describes how to install and use the Altiris® Streaming Composer to create packages, for use with the Altiris® Streaming System.

Assumptions

To make the best use of this guide, we assume you are familiar with the following concepts:

- The structure of the applications you need to package and stream.
- Application streaming technology.

Typographic Conventions

The following list shows the special type styles used in this guide.

<code>code</code>	Computer output and file names
user input	Code or commands you type (bold)
<i>user input,variable</i>	Code or commands for which you determine the value and type (bold and italic)

In this Guide

This document is organized as follows:

Chapter 1: Streaming Composer Overview—Provides an introduction, installation information, and system requirements.

Chapter 2: Using the Streaming Composer Interface—Describes the graphical user interface and how to use it.

Chapter 3: Process Walkthrough—Provides an overview of the basic steps required to create a package.

Chapter 4: Customizing a Package—Describes how to use the Streaming Composer tools in the Tools menu to fine-tune packages.

Chapter 5: Creating a Snapshot Package—Describes how to create a Snapshot package in detail.

Chapter 6: Creating an MSI Package—Describes how to create an MSI package in detail.

Chapter 7:Creating a Tracking Package—Describes how to create packages for tracking applications, conventionally installed on client computer desktop.

Chapter 8:Creating a Package for SVS Applications using a .VSA file—Describes how to create packages for virtual applications i.e. .VSA.

Chapter 9:Upgrading a Package—Describes how to upgrade a package.

Chapter 10:Using Command-Line for Packaging—Describes how to use Command-line to create/upgrade a snapshot package.

Appendix A : Execution Script-An Example

Appendix B: AppstreamCfg File

Appendix C : Packager.INI File

Appendix D : Appstream.INI File

Appendix E : How to Deploy Service Packs & Hotfixes

Chapter 1

Streaming Composer Overview

The Streaming Composer creates packages for streaming. A package may contain multiple applications. It holds all the files that make up the entire set of applications.

The Streaming Composer identifies the files you need to install for the successful execution of the application and breaks it into streamlets. It creates a package that you can upload to an Altiris Streaming Server.

The Streaming Composer supports only 32-bit applications. You can create the following types of packages :

- **Snapshot Package**—A Snapshot package exposes the executables for the applications. Therefore, you can create a package with one application or with a suite of applications.
- **MSI Package**—Applications created with Windows Installer/MSI should only be packaged as a MSI package. The logic available with the application installer is used to set up the entire system at the time of streaming the application. The entire package is provisioned.
- **Tracking Package**—Tracking packages enable tracking the usage of applications that have been conventionally installed on client computer desktop. By default, they are offline access enabled.
- **SVS Package**—Some applications can be virtually installed on the client computer desktop using the virtualization facility offered by Altiris. The Streaming Composer supports package creation of .vsa files created using Altiris. Such packages can be subsequently streamed and monitored for license usage.

The Streaming Composer is a powerful tool. An easy-to-follow user interface takes you through the packaging steps for each type of package.

System Requirements

The Streaming Composer requires the following hardware and software components. To create a package, the system must be clean i.e. the system should only be loaded with the operating system and the minimal utilities required to run the application. Alternatively, the system can be a mirror image of the system where the application may ultimately run.

Hardware

- System Processor : Pentium III, 700 MHz
- System Memory : 256 MB memory minimum (512 MB recommended)
- Free Disk Space : 30 MB of free disk space, plus three times the size of the application package

Software

- Windows XP
- Windows 2000
- Windows 2003 Service Pack 1
- Windows Vista

Prerequisites

System must be clean i.e. the system should only be loaded with the operating system and the minimal utilities required to run the application. Alternatively, the system can be a mirror image of the system where the application may ultimately run.

Installing the Streaming Composer

The Installation CD can be used for the installation process. When this CD is inserted into the CD-ROM drive, an installation menu pops up.

1. Navigate to **Install Products>Install Streaming Composer** option to install the Streaming Composer on your system.

Note : If the installation wizard does not start automatically after you insert the CD, open `\Packaging\PackagerSetup.exe`. Click the `.exe` file, to install the Streaming Composer. If your system is working on Windows Vista, then it seeks your permission to run `PackagerSetup.exe`. Click **Continue**.

2. The Installation wizard opens. The installation program scans your system and determines whether it has enough memory to accommodate the application. On the Setup window, click **Next**.
3. The License Agreement is displayed. Read the agreement. Click **Yes**.
4. On the Destination Folder window, click **Next** to accept the default installation location (`[current Windows drive:]\Program Files\Altiris\Streaming Composer`). If you want to use a different folder, click **Browse**. You can browse to designate a different folder.
5. Click **Next** in the Select Program Folder window to continue installation with default values.

6. By default, the License File Selection window, displays the location of the trial licence, that comes with the system. If you have bought a license, then click **Browse** to locate the license file (*.lic). Click **Next**.
7. Click **Finish** to complete setup.
8. The Installation program installs the Streaming Composer on your system and creates a shortcut in the Windows Start menu.

Summary

This chapter has given a preliminary overview of the Streaming Composer, the hardware and software requirements required for installing the Streaming Composer and the actual Installation procedure.

The next chapter describes the Streaming Composer user interface. This interface guides you through the entire packaging process.

Chapter 2

Using the Streaming Composer Interface

The Streaming Composer user interface guides you through the packaging process.

The Streaming Composer interface includes the following elements:

- **Menu Bar** : File, Edit, Tools, and Help menus.
- **Tool Bar** : Toolbar buttons to facilitate common tasks.
- **Navigation Pane** : A tree view of all packaging steps. Identifies steps you have completed and those you need to complete.
- **Primary Pane** : Displays online documentation and data entry pages.
- **Status Pane** : Displays next step(s) you need to complete and a log view.

Using the Menu Bar

The Streaming Composer Menu Bar contains the following menus: File, Edit, Tools, and Help. This section describes each menu item.

File Menu

Use this menu to carry out different operations on a package.

File Menu Item	Function
New	Creates a new package.
Open	Opens an existing package.
Save	Saves current settings. The Streaming Composer automatically saves most settings when you move to another step.
Close	Closes current package and saves current settings.
Exit	Exits from the Streaming Composer.

Edit Menu

Use this menu to reset and clear log, page and environment content.

Edit Menu Item	Function
Clear Log	Clears the Streaming Composer log.
Reset Current Page	Resets all values on the current page and reverts to the last saved package environment content.
Reset Environment	Resets the environment and reverts to the last saved package environment content.









Tools Menu

Use this menu to set Configuration Profiles, Packaging Options, and Preferences for a package. These are used to customize a package that is being created by Streaming Composer.

Tools Menu Item	Function
Configuration Profiles	Sets up a Configuration Profile that you can re-use for different packages. When you first create a package, you can choose a standard Configuration Profile that the Streaming Composer provides or a custom Configuration Profile.
Packaging Options	Defines rules for Snapshot and Packaging processes. The rules define how the engine behaves when it handles files, INI files, and registry entries. The Streaming Composer uses these rules when it creates the package.
Preferences	Specifies a repository location for your packages. You can also specify the Streaming Console URL to upload packages to the streaming system.
Backup Package	Backup the current versions to a different location.
Install Package	Allows the system administrator to install or restore the existing Snapshot package without using the original installation disk.

Using the Toolbar

The following table describes the icons in the toolbar buttons and their function. You can also hover the mouse over each icon to bring up a ToolTip that describes the function of the icon.

Icon	Function
	Creates a new packaging project.
	Opens an existing package.
	Saves the current package.
	Moves to the next step in the packaging process.
	Resets the values on the current page.
	Updates the package environment with changes from the file, registry, and INI pages.
	Opens the Engine Options dialog box.
	Displays an About box.

Using the Navigation Pane

The Navigation Pane contains a tree list of all the steps required to package an application. As you use the Streaming Composer, you notice that the arrows in the Navigation Pane lead you from one step to another. A green arrow indicates that the action must be com-

pleted, and a green checkbox indicates a completed action. A red “X” mark indicates an uncompleted step. A red “X” appears if you change a variable that affects a step you have already completed.

At times, the Streaming Composer performs several steps in the background. The arrow then skips several steps and settles on the next step that requires user action. You can also go back to the previous step and redo the action, if required.

Using the Primary Pane

The Primary Pane displays all the information related to packaging an application. This pane displays a dialog box that contains additional configuration parameters. The arrow in the Navigation Pane shows the step you are working on in the packaging process. The step also appears at the top of the page. You can also navigate to certain previous steps and redo the process.

The Primary Pane also contains online documentation. Each phase heading includes an overview of the packaging process for that phase. Click a phase heading on the Navigation Pane to view additional information about a step.

Using the Status Pane

The Status Pane provides a text link to the next step required to complete the package. The Status Pane is visible at lowermost end of the window. The two tabs available are:

- **Assistant** : Displays the next step to be activated for completing the packaging process.
- **Log** : Click this tab to read the log generated by the Streaming Composer during the packaging process.

Summary

This chapter describes the Streaming Composer Interface and helps you navigate through the various options provided.

The next chapter describes the built-in tutorial that guides you through the various phases of the packaging process.

Chapter 3

Process Walkthrough

This chapter walks through the different phases involved in the creation of a package. The Streaming Composer recognizes Snapshot packages, MSI packages, Tracking Packages, and SVS packages. Some of the processes to be followed are different for each of them. The Streaming Composer uses the following basic phases to create a package:

- *Creating a Package*
- *Reviewing Package Content*
- *Building a Package*

The following sections describe each of these phases and also how to get started with the Streaming Composer.

Creating a Package

The package creation steps are different for a Snapshot and a MSI package.

- When you create a Snapshot package, the Streaming Composer takes a snapshot of the system, installs the application, takes a second snapshot, and compares the two snapshots. It processes the differences between the two snapshots.
- When you create a MSI package, the Streaming Composer imports all the files and logic of the MSI installation. The original MSI setup logic determines the location of the files and logic at run time.
- When you create a SVS package, the Streaming Composer imports the VSA environment.
- When you create a Tracking package, the Streaming Composer only requires the basic details of the package to be created.

Reviewing Package Content

The Streaming Composer displays the Program and Data files, Registry entries, INI configuration files, and the Environment Variables which the application altered or added to the system during installation. You can evaluate and change each one as needed.

Note : In MSI packages, the Package Manifest page gives information about the contents of the installed application and is a read-only page. Refer “*Reviewing the Content for an MSI Package*” on page 45, for more information on Manifest elements.

Updating Package Information

You can customize the following:

- Modify properties and set system requirements
- Set and configure execution scripts
- Modify and remove execution scripts
- Modify or define new applications
- Create a startup block for a snapshot application

Building a Package

The Streaming Composer analyzes the content of a package, and creates the application streamlets for the Streaming Agent. Depending on the size of the package, it creates a ZIP file that contains the required package files. You can now use the Streaming Console to upload and provision the package to users.

- Refer to “*Building the Package*” on page 38 for more information on building a Snapshot package.
- Refer to “*Building the Package*” on page 49 for more information on building a MSI package.
- Refer to “*Package Building Phase*” on page 54 for more information on building a Tracking package.
- Refer to “*Building the Package*” on page 56 for more information on building a SVS package.

Getting Started

Start the Streaming Composer using the Windows Start Menu. The opening window shows the Streaming Composer interface described in the previous chapter. You can either create a new package or modify an existing package. The details of the packages created so far, are all available in a Repository.

Using a Repository

The Repository contains a list of all the packages you have created. You must configure the repository in the Preferences page on the Tools menu if you want it to be active and point to the correct folders.

The Repository fields are read-only. The data in the fields is read from the information you entered, when you created the package.

Note : Make sure that you open and save packages from a local storage device, and not from a network storage device. Use drive mapping for the network storage device.

The Repository provides the following information about each package:

Repository Field	Definition
Directory	The location of the package.
Name	Name specified for the package.
Type	Package type: MSI or Snapshot.
Version	Version number specified by the Streaming Composer.
Created Time	Time of creation of this package.
Modified Time	Time of modification of this package.
Vendor Name	Name of the application developer or publisher.
Vendor Version	The version number assigned to the application by the vendor.
Packaging Version	Version of the Streaming Composer used to create this package.

Using the Streaming Composer Window

When you start the Streaming Composer:

- The Welcome window with the menu bar is displayed.
- The Streaming Composer dialog box is also displayed.

You can choose to create a new package or open existing package by selecting the suitable tab.

Alternatively, you can click the **New** option from the File menu to create a new package or select the **Open Existing** option from the File menu to open an existing package.

To Open an Existing Package

Existing packages can be opened to modify them or to check their parameters.

1. In the Open Existing tab page click the three-ellipses button in the **Package Path** field to browse for a package. Click **OK**.
2. Click **Repository**. Double-click a package in the repository. The package location populates the **Package Path** field in the Open Existing dialog box. Click **OK**.

Note : Define the repository path using the Preferences option. If not, an error message is displayed. Refer “*Defining the Preferences*” on page 23 for details on Preferences.

Note : To modify a package, the system must be in the same state as it was when you created the original package. If it is not in the same state, a message recommends that you open the package in read-only mode. Make sure that you open and save packages from a local storage device, and not from a network storage device. Use drive mapping for the network storage device. If some of the files in the root path of the package are deleted, then the Streaming Composer displays a error message and recommends that you open the package in read-only mode.

Conventional Package Installation

This feature allows the system administrator to install or restore an existing Snapshot package without using the original installation disk

Converting Older Packages

To stream applications using Streaming Composer version 5.2.2 server, it is essential that packages conform to the version 5.2.2 format.

Older packages - both Snapshot packages and MSI packages - created using the Streaming Composer versions older than 5.0, need to be converted to adhere to the version 5.2.2 format. When you use Streaming Composer version 5.2.2 to open an older package, the system prompts you to convert to the new package format. Click **Yes** to automatically convert the package (s) to the latest format.

Note : Once these packages are converted, they cannot be used from Streaming Composer 4.0. Hence it is recommended to backup the package before attempting to convert it to the new format.

Convert the old MSI packages

Follow the procedure given below, for packages created using Streaming Composer versions older than 5.0.

1. Launch the new Streaming Composer (build 5.2.2) or later.
2. Open the old MSI package in this version.
3. Navigate to **Tools>MSI Handling>Non-Nullified Files** from the menu. On the Non-Nullified page change `*\sku*.xml`, `*\license.txt`, `*\instmsi*.exe`, `*\system32*`, and `*\Documents and Settings*`.
4. Select Import MSI page, click **Advanced** button and execute all the tasks.
5. Change the executing script and setup configuration, if necessary.
6. Continue to the next page and create streamlets and package.

! The Streaming System now supports services by default and no extra scripting is required to achieve the same. So, all packages that were created with special scripts for streaming services need to be repackaged without those scripts.

Summary

This chapter describes the different phases of creating a package. It also describes the Repository and its use. The next chapter describes tools provided in the Streaming Composer to customize packages.

Chapter 4

Customizing a Package

The Streaming Composer provides tools to customize a package on a granular level. You can find most of these tools in the Tools menu on the Streaming Composer menu bar. The Tools menu contains the following sections:

- *Using Configuration Profiles*
- *Defining Packaging Options*
- *Defining the Preferences*

Using Configuration Profiles

The Configuration Profile stores the packaging parameters, including Snapshot monitoring and engine parameters. Configuration Profile minimizes the need to change engine parameters when you switch from one application to another. You can select the appropriate Configuration Profile for each application type. You can create multiple Configuration Profiles, but only one is active at a time.

Set Global Configuration Profile

Changes to the Configuration Profile at the global level affect all new packages that use this profile. The Global Configuration Profiles include:

- **A `exclude_setup_info` configuration**—This configuration has the same values as the minimum configuration, except that it excludes any existing Windows Installer files and registry entries inside the package content.
- **A `default` configuration**—This configuration has common values that you can apply to most applications.

To modify the global Configuration Profile parameters, navigate to **Tools>Packaging Options**, when a package is not loaded. Never modify the `exclude_setup_info` or default configuration profiles. Always create a clone.

When a new package is created, the Streaming Composer makes a copy of the global configuration profile, places it in the package folder, and renames it. For example, the name of a package that uses the default global configuration profile in the folder- `c:\packages\MyFirstPackage` is `MyFirstPackage-Default`.

When you create an upgrade package, the Streaming Composer copies the original active configuration profile to the new upgrade package. For more information about upgrading a package, refer to the section “*Upgrading a Package*” on page 59.

To Set an Active Configuration Profile

1. Navigate to **Tools>Configuration Profiles**.
2. Highlight a configuration profile from the list displayed. Click **Set Active**.
3. To use the configuration profile as is, click **OK**.
4. To modify the configuration profile, navigate to **Tools>Packaging Options**.

To Rename a Configuration Profile

1. Navigate to **Tools>Configuration Profiles**.
2. Highlight a configuration to be renamed from the Configuration Profile list. Click **Rename**.
3. Enter a new name for the profile in the empty field. Click **OK**.
Note : You cannot rename the Default Configuration Profile.
4. The Configuration Profile list is updated with the new name and the date it was last modified. To use the renamed profile, highlight it and click **Set Active**.
5. Navigate to **Tools> Packaging Options** to define parameters for the new profile.

To Create a Configuration Profile Clone

1. Navigate to **Tools>Configuration Profiles**.
2. Highlight a configuration to clone from the Configuration Profile list. Click **Clone**.
3. Enter a new name for the clone in the empty field. Click **OK**.
4. The Configuration Profile list updates with the new clone and the date it was last modified. To use the clone, highlight it and click **Set Active**.
5. Navigate to **Tools>Packaging Options** to define parameters for the new profile.

To Remove a Configuration Profile

1. Navigate to **Tools>Configuration Profiles**.
2. Highlight a Configuration Profile from the Configuration Profile list. Click **Remove**.
3. Click **Yes** to remove the configuration profile.
4. Click **OK**.

Note : Default and Exclude_Setup_info cannot be deleted.

Defining Packaging Options

Use the Packaging Options to specify parameters for the Streaming Composer. The Packaging Options section specifies how the Streaming Composer monitors drives, folders, registry hives, and file version information.

Navigate to **Tools>Packaging Options**. The Packaging Options menu contains the following sections: Monitoring, File Handling, Registry Handling, INI File Handling, MSI Handling, and Build Processing.

Using the Monitoring Group

The Monitoring Group folder contains three sections: Files, File Versions, and Registry Hives.

Monitored Files

Use the Monitored Files pane to add, modify, or remove drives and folders that you want to monitor. By default the drive to be monitored is set to the current system drive.

The streaming system recognizes several variables that represent some of the more commonly used paths. Click **Variables** to view a list of all variables and paths. Use these variables to specify path parameters in all the Packaging Preferences screens.

File Versions

Use the Monitored File Versions pane to monitor file versions. You can add, modify, or remove file extensions that need to be monitored. The Streaming Agent uses file version information when it installs files outside of the cache to make sure that the latest version is installed.

Registry Hives

The Monitored Registry Hives pane displays a list of registry hives for the Streaming Composer to monitor. You can select a check box, to monitor a registry hive.

The default hives include: HKEY_LOCAL_MACHINE, HKEY_CURRENT_USER, and HKEY_CURRENT_CONFIG.

Note: You do not need to specify the HKEY_CLASSES_ROOT hive, since it is included in the HKEY_LOCAL_MACHINE hive.

File Handling

File Handling Options allows you to add or remove excluded files, installable files, mark them as dynamic files or static files for the engine to process. The following section describes each of these types and provides examples of how and when to use them.

Excluded Files

Use the Exclude option to exclude files from the package. The file earmarked for exclusion is not installed in the client computer when the package is streamed. You can use wildcard characters and certain variables to exclude files and folders. For example,

- To exclude all the sub-folders and files under the folder `c:\windows\system32\config`, designate the folder as `$(systemdir)\config*` and the file as `*`.
- To exclude all the files with the extension `.lnk`, specify the folder as `*`, and the file as `*.lnk`.

To view the different variables, click **Variables** when you add or modify an item.

Installed Files

Installable files are included in the package, but not streamed. Installable files are installed outside the Streaming Agent cache during virtual installation. Installable files include Windows system files and files in user folders.

You can use wildcard characters to specify file and folder information. For example, make all files in the `$(windir)` installable, specify the folder as `$(windir)*` and the file as `*`.

Dynamic Files

Dynamic files are text files that contain references to files within the package. The Streaming Composer processes these references to point to the correct location. Mark text files, such as XML or application specific text files, as dynamic.

Use wildcard characters to specify file and folder information. All files with the extension `.bat` and `.cmd` are dynamic by default. For example, to make all files with the extension `.xml` dynamic, specify the folder as `*`, and specify the file as `*.xml`.

Static Files

Static files are not modified by the Streaming Composer. Use this option when you do not want the Streaming Composer to modify a file, that it would normally treat as a dynamic file and modify. You can use a `*` wildcard character to specify all the static files in a folder.

Registry Handling

Registry Handling provides two options:

- **Excluded** : To exclude designated registry entries that must not be part of the package.
- **Ignored** : To specify designated registry keys and values that must be ignored during the conversion process.

Excluded Registry Entries

Excluded Registry Entries include system specific registry entries and also temporary, irrelevant keys and values. The entries earmarked for exclusion are not installed in the client computer when the package is streamed. Use wildcard characters to specify keys and values.

For example, to exclude all subkeys and values from the key `HKEY_LOCAL_MACHINE\SYSTEM\MOUNTEDDEVICES`, you need to specify the following key:

```
HKEY_LOCAL_MACHINE\SYSTEM\MOUNTEDDEVICES* with the value "*".
```

Ignored Registry Entries

Ignored Registry Entries include registry entries from the conversion process that might cause the application to run improperly, if converted. By default, the Streaming Composer changes the values of the Registry Entries. If your package needs to retain the values as it is, then mark this entry as an Ignored Registry Entry.

The Streaming Composer uses variables to convert specific path information to an independent form. Path conversion is critical for the correct operation of the package on different desktop environments.

For example, the following string contains a reference to `C:\program files` as well as to a different system. If the following string is converted, it may cause a problem when you stream the application:

```
System-A\C:\program files\MyProgram|LicenseFolder|LicenseFile.lic
```

INI File Handling

Use INI File Handling to specify excluded, ignored, non-parsed, and parsed INI files in the conversion process.

Excluded INI Files

Exclude system specific, temporary, and other irrelevant INI files and sections from the package because they might cause the application to run improperly. You must exclude the complete section, not values from the section.

You can use wildcard characters. For example, to exclude all sections of the file `DESKTOP.INI`, specify the INI file as `*\DESKTOP.INI` and the sections as `"*"`.

Ignored INI Files

The Streaming Composer uses variables to convert specific path information to an independent form. By default, Streaming Composer changes the values in the INI files. If your package needs to retain the values as it is, then mark this file as an Ignored INI File.

In most cases, path conversion is critical for package to operate correctly. However, if an application requires a license from a remote system and the file is read from an INI file, do

not specify the file for conversion. For example, to ignore the section `LicenseServerPath` from the file `MyAppConfig.ini`, you need to specify the INI file as `*\MyAppConfig.ini` and the section as `[LicenseServerPath]`.

Non-Parsed INI Files

Non-Parsed INI files include files that do not comply with Microsoft standards for an INI file. If the Streaming Composer encounters a non-parsed INI file, it keeps the file as a text file and does not parse it.

Use wildcard characters to specify file information. For example, to mark all `config.ini` files in the `C:\Program File\AutoCAD` folder as non parsed file, you need to specify the file path as `C:\Program File\AutoCAD*` and file as `config.ini`.

Parsed INI Files

Parsed INI files are files that comply with Microsoft standards as an INI file. By default, all INI files are parsed unless they are specified in the non-parsed list.

Use wildcard characters to specify file information. For example, to mark the section `LicenseServerPath` in the file `MyAppConfig.ini`, you need to specify the INI file as `*\MyAppConfig.ini` and the section as `[LicenseServerPath]`.

MSI Handling

The MSI Handling options apply to non-nullified files, or files that the Streaming Composer must not process. Use this option to select files that should not to be nullified.

Build Processing

An orphan path does not exist in the package, but has path references in the registry, INI files, or in the dynamic files. Use Build Processing to specify how the Streaming Composer should handle orphan path references in registry, INI, or dynamic file references, as follows:

- Specify orphan paths as referenced to an installed file or referenced to a streamed file.
- Find and replace specific strings in dynamic files, registry, and INI files. Specify strings when the package requires the new path for correct operation, but the Streaming Composer does not automatically convert the path.
- Specify strings that the Streaming Composer ignores when it builds the package. These strings preserve the integrity of the original application content. The Streaming Composer does not convert the list of ignored strings.

Note : Settings in this section are global and apply to INI files, dynamic files, and registry entries throughout the system. Exercise caution when you modify global options.

Orphan Streamed References

Specify the path reference for the streamed file. The file referenced by this path does not exist in the package. This could be created at run time or currently exist elsewhere on the system. For example, the following file reference is not part of the package and did not exist on the system after the post-setup Snapshot:

```
C:\Program Files\My Application\Log\Application.log
```

Specify this reference to a streamed file because `C:\Program Files\My Application` must be in the cache since it is the installation location for all the application files.

Orphan Installed References

Specify the path reference to the installed file. The file referenced by this path does not exist in the package. This could be created at run time or may currently exist elsewhere on the system. For example, the following file reference is not part of the package and did not exist on the system after the post-setup Snapshot:

```
C:\TestLib\test.dll
```

Specify this reference to an installed file because `C:\TestLib` may be a location for a third-party library used by the application and hence reside outside of the cache.

Ignored Content

Ignored content includes string patterns of entries that the Streaming Composer does not convert. If the Streaming Composer finds these strings in any of the registry, dynamic files or INI values, then it does not convert the entry.

This string is not case sensitive and you can use wildcard characters. To exclude the conversions of all the registry and INI values that contain the string `system-A`, specify in the pattern field `*system-A*`.

Replaced Content

Specify the content in registry, dynamic files, or INI values that you want to find and replace.

Note: The content of a Find string cannot be part of a Replace string. For example, to replace a current user's desktop path to streamed user desktop path in INI or Registry, you need to specify `C:\Documents and Settings\Administrator\Desktop` as the find string and `$userdesktopdir$` as the replace string.

Defining the Preferences

The Preferences section lets you configure the display of the steps in the packaging process, and the storage location of the packages you create.

You can perform the following:

- Click the check box next to **Automatic Next Step**, if you want the Streaming Composer to automatically move to the next highlighted step. This is the default setting.
- Browse for a directory path. This is the default location for your package repository. Use the repository to view a list of all the packages you have created.
- Type the location of the Streaming Console URL. The **Launch Streaming Console** button on the Build phase reads this location and enables you to upload a package to the Console through the Streaming Composer.

Summary

This chapter has discussed the different parameters that can be customized prior to packaging an application.

The next chapter gives details of the procedure to be followed to create a Snapshot package.

Chapter 5

Creating a Snapshot Package

This chapter describes how to create a Snapshot package. The following sections describes the procedure in detail.

- *The Create Phase*
- *The Package Content Phase*
- *The Package Information Phase*
- *Building the Package*
- *Best Practices for a Snapshot Package*

The Create Phase

Use the following method to create a package:

1. Take a baseline snapshot.
2. Install the application.
3. Take a post-setup baseline snapshot.
4. Process the difference.

In most cases, create a package on a clean system because you can then install the application to most desktop systems. However, if each computer that you plan to install the package on has the same image, then create the package on a system with the common image.

The Create phase consists of the following steps:

- Name the package and determine its location. The recommended location is the package repository. (Refer “*Using a Repository*” on page 12 for more information.)
- Take a baseline snapshot of the system. This system mirrors the system where the application may run.
- Install the application on the system.
- Take a post-setup snapshot of the system.
- Process the difference between the system snapshots, without and with the installed application.

To Create a Package

1. Navigate to the Streaming Composer on the Windows Start menu: **Start>Programs>Altiris>Streaming Composer**. Execute the application.
2. Click the **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
3. On the Create New page displayed, choose the **Snapshot Package** option.

The completed package resides in the **Package Path** folder. Enter the folder path or browse to the designated package folder. This path should not be the same as that of the path of the current operating system. It must be complete and cannot be a network path. For example, `D:\myPackages\PaintShopPro`.

4. The **Package Name** is the name you assign to the package. Enter a name.

! Do not include special characters like “&” in the package name.

Note : Zip files cannot be created for packages with Unicode names. So while packaging such packages, set the flag `packager.build.zip.skip=true` in `packager.ini`. This setting forces the Streaming Composer to skip the step to zip the packages with Unicode names.

5. Choose one of the two global configuration profiles that the Streaming Composer provides: **Exclude_Setup_Info** or **Default**.
 - **Exclude_Setup_Info**—This configuration has the same values as the minimum configuration, except that it excludes any existing Windows Installer files and registry entries inside the package content.
 - **Default**—This configuration has common values that you can apply to most applications.

(or)

You can choose to use any of the profiles already available in the system. Browse to the required folder and select the appropriate profile.

(or)

You can also create a custom global configuration profile. To view or modify the default values, see the entries under **Tools>Configuration Profiles**. The Streaming Composer places a copy of the configuration profile in the newly created package folder. This configuration profile is now active for this package. To learn more about configuration profiles, refer to the section, “*Using Configuration Profiles*” on page 17.

6. Click **OK** and the Create Application screen is displayed, to take you through the steps of creating the snapshot package.

Note : When you log on with a view to creating a new package, ensure that the user name and computer name are significantly different. If these are not different, while creating a

new package you see an error message - *To avoid packaging problems, user name and computer name should be significantly different.*

! Do not log on as Administrator, while creating a package.

To Take a Baseline Snapshot

1. The **Baseline Snapshot** step records the state of the system before you install the application.
2. Click **Baseline Snapshot** under the **Create** menu in the Navigation pane. Take or Import Baseline Snapshot screen is displayed.
3. Click **Run** to take the snapshot. The Streaming Composer scans the file system for relevant files and registry entries. This process may take a few minutes.
4. You can also load a snapshot information (taken during a previous packaging process done on this machine) from disk. Click **Import** to load the snapshot. Ensure that the imported snapshot is identical to the state of your current system, else the final package may not run properly. At any point of time, you can use the Import option to load the baseline or the post-setup snapshot file created previously.

To Install the Application on a System

1. The **Application Setup** page is displayed. The Navigation Pane and the Status Pane display this step. Skip this step if you plan to install the application manually.
2. Enter the complete path of the setup file for the application or use the three-ellipses button to browse for it.
3. If you select a Windows Installer setup file, we recommend the use of command-line parameters.
4. Enter any command-line options required to run the setup program.
5. Click **Run** and the application is installed.

Note : Take an image of the system after the installation of the package, if you need to upgrade this package later. You will be prompted with a list of recommended command-line parameters for this setup. Click **OK** on this dialog. On completion of the installation, do not invoke the application, since some applications generate files/folders when invoked for the first time.

To Take a Post Installation Snapshot

1. The **Post-Setup Snapshot** page is displayed. The Navigation Pane and the Status Pane display this step.
2. Click **Run**. The Streaming Composer scans your file system for relevant installed objects and takes the post-setup snapshot of the files, registry entries after the applica-

tion is installed. The length of the scan depends on the number of files and registry entries on the system or path.

To Process the Difference Between Snapshots

In most cases, when you process the difference between the snapshots, the package meets all requirements and you can continue to examine the package content. You see a dialog box pop up, if one of the following events occurs:

- **INI file handling**— By default Streaming Composer has been configured to carry out this process automatically. If you wish to carry out this process manually, set the parameter `packager.engine.showparse` to `true` in `Packager.ini`.

If the parameter has been set to manual, when the Streaming Composer encounters INI files that do not conform to Microsoft standards, it displays a dialog box that allows you to determine how the Streaming Composer manages non-conforming files. Select a file and click **Details** to see the contents of the file. Each non-conforming line in the INI file is red. Determine if the Streaming Composer should parse the content as an INI file or keep it as a static binary file.

- **Control Panel File**—When the Streaming Composer encounters a control panel file (`.cp1`) as it processes the difference between the snapshots, it displays a dialog box that suggests a shortcut path for the `.cp1` file. The `.cp1` file may run incorrectly if it is not launched by an streaming shortcut. Use the dialog box to modify the name of the shortcut and the location of the shortcut path on the client computer system. Click **OK** to add the shortcut or click **Cancel**.
1. The **Process Differences** page is displayed.
 2. Select or enter the correct **Root Path** for your application. The Streaming Composer streams files that are on the Root Path directory or its subdirectories to the Streaming Agent cache. Files that are not installed on the client computer computer by the Streaming Composer, remain outside the cache.
 3. Click **Run**.

The Package Content Phase

The Package Content phase displays manifest information, program and data files, registry entries, INI configuration files, and environment variables that the application has altered or added to the system. It also applies any changes that you specified to the active configuration profile. Use this phase to review these elements and change any that do not belong in the package.

Manifest Information

The **Package Content>Manifest Information** displays a high-level read-only summary of elements from the pre- and post-snapshot comparison. The Package Manifest Informa-

tion displayed includes Windows file protection files, services, control panel files, ODBC data sources, fonts, files association and startup executables set by the application.

A green check mark represents contents that are supported by the Streaming Composer and a red cross mark represents contents that are not supported directly by the Streaming Composer.

The manifest information includes:

Manifest Element	Description
Windows File Protection	Windows file protection files in the package. This version of Streaming Composer does not support the installation of WFP files.
Services	Services installed by the application. This version of Streaming Composer supports services like Windows Services etc. Some of the services and drivers like VPN client services are not yet supported by Streaming Composer.
Control Panel File	Control panel files in the package.
ODBC Driver	ODBC drivers in the package.
ODBC Data Source	ODBC data sources in the package.
Font	Fonts in the package.
File Association	File extensions associated with the package.
Startup Executable	Executables that automatically run after log in. This feature is not supported in this version.

Note : Streaming Composer supports packages which require installation of drivers. However it is essential that these drivers are not installed or streamed from within the Streaming Agent Cache folder. Hence, if a driver needs to be installed along with the streamed package, ensure that you specify (while packaging) that the driver be installed outside the cache.

Review File, Registry, and INI Files

Package Content>Files, Package Content>Registry and Package Content>INI display the program and data files, the registry entries, and the INI configuration files that the application has altered or added to the system during installation. The Streaming Composer automatically excludes unnecessary registry entries in Snapshot packages. However, it is recommended that you review the package contents to exclude unnecessary file, registry, and INI entries that are not part of the application. The File, Registry, and INI file steps work in the same way, with the exception of the action buttons. Different action buttons appear for different entry types as follows:

- Package files display all action buttons.
- Registry entries display **Exclude/Include** buttons.
- INI entries display all action buttons except **Dynamic/Static** buttons.

To Review Files, Registry Entries, and INI Entries

1. Navigate to **Package Content>Files** or **Package Content>Registry** or **Package Content>INI** as the need be.
2. Select an entry from the drop-down list.
3. Use the tree view to navigate through the elements. As you select each element, the **Files, Registry entries, or INI files** associated with it are exposed in the lower table. The **Path** field shows your selected node.
4. In the lower table, select the file, registry entry, and INI file and click one of the following buttons to perform an action on it. You can set more than one attribute. The **AS Attributes** column lists the current state of each entry. Hence **S** in this column indicates that the Stream attribute has been selected. The valid attributes are:
 - **Install/Stream**—Install as denoted by the letters **I**, and Stream is denoted by the letter **S** in the **AS Attributes** column. If you change a streamed entry to an installable entry, it resides outside of the Streaming Agent cache. If you change an installable entry to a streamed entry, it streams to the Streaming Agent cache.
 - **Exclude/Include**—Exclude is denoted by the letter **E** in the **AS Attributes** column. Change an excluded entry to be included in the package or change an included entry to be excluded from the package.
 - **Delete/Keep**—Delete is denoted by the letter **D** in the **AS Attributes** column. Delete an entry from the package and instruct the Streaming Agent to delete it from the cache if present, or keep an entry that was already deleted. Use this feature when upgrading a package.
 - **Dynamic/Static**—Dynamic is denoted by the letter **X** in the **AS Attributes** column. A dynamic file is parsed by the Streaming Composer, to find any known application fixed path references and replace them with variables. During virtual installation, the Streaming Agent replaces the variables with relevant references to the application in the Streaming Agent cache. A static file maintains its original content. The Streaming Composer does not parse a static file.
 - **Overwrite/Auto** —Overwrite is denoted by the letter **O** in the **AS Attributes** column. The file is overwritten, irrespective of the automatic upgrade algorithm.

Environment Variables

When you navigate to **Package Content>Environment variables**, the screen displays a list of environment variables that have been modified or added to the system by the packaged application. You can modify or remove the environment variables from this list or add new environment variables to it.

Environment variables are a set of dynamic values that decide the processes at the runtime. The data stored in these variables help you earmark your home directory, your preferred text editor, or the directories you would like to check while running programs. A sample list of valid variables:

Windows Environment variable

`%SystemDrive%` C:

`%SystemRoot%` C:\WINNT C:\WINDOWS

`%HOMEPATH%` The path to the user's home directory (excluding drive): \Documents and Settings\

`%USERNAME%` The user's name

Environment Variable

`$APPS_PATH$` Streaming Agent cache (c:\AltirisCache\fltroot\xxxx\)

`$systemdir$` C:\WINDOWS\SYSTEM32

`$programfilesdir$` C:\PROGRAM FILES

`$userprogrammenudir$` C:\DOCUMENTS AND SETTINGS\ASUser\START MENU\PROGRAMS

`$windrive$` C:\

Package Environment

Package Content>Package Environment rebuilds the package environment, and saves your changes to the Package Content. If you change any files, registry entries, or INI files in this phase, then you must rebuild the package. Click **Update** to rebuild the package.

The Package Information Phase

Use the Package Information Phase to update package properties, set system requirements, define execution scripts and applications, split the application into streamlets, and collect data for the startup block.

Define Properties

Use **Package Information>Properties** option to specify or update additional package properties. The following table describes the fields on this page:

Package Property	Definition
Package Path	The location where the package is stored.
Original Package Path	The location of the original package, when upgrading a package.
Package ID	The unique identifier assigned by the Streaming Composer.
Source Path	A folder under package path in which empty folder structure of the application is created. This is enabled only if you select Create Source Folder .
Package Name	The package name specified during the Create phase. Used by the Streaming Agent and the Streaming Console to specify the package. You can change this name at any point, during the packaging process.
Package Version	The package version number assigned by the Streaming Composer. The version number is 1 for new packages. This number increments for each new upgrade version. Used by the Streaming Console and the Streaming Agent to distinguish different versions of a package.
Vendor Name	The name of the application developer or publisher.
Vendor Version	The version that the developer or publisher assigned to the application.
Installation Type	A free-text field that describes the type of media used to install the application.
Comments	A free-text field to document any changes made to the package during the build process. The information in this field becomes part of the build <code>readme.txt</code> file. This information is useful to anyone who needs to rebuild the package.

Specify System Requirements

Use **Package Information>System Requirements** to specify the operating systems that support this package. This information is used by the Launch Server to ensure that the access is restricted to only those provisioned applications that run on the operating system currently running on the client computer. Check the **Requires reboot after installation**

check box, if you want the client computer to reboot the system after installing the application.

Define Execution Scripts

Package Information>Execution Scripts is used to define scripts that the Streaming Agent should execute for some milestones (like pre- installation, post-uninstallation etc.) for an application. Click **Add** to open Defining Execution Scripts screen.

An execution script is an executable file, such as a batch file. The Streaming Agent invokes the script with command-line parameters and examines the result on completion of the execution. The application does not run if the execution script fails. You can create multiple execution scripts for an application. They can be designed to be invoked during different phases of the application installation and execution as explained below.

For example, you can use an execution script to ensure that the client computer system has the required environment to execute the application. Some scripts verify the existence or absence of third-party components or certain application versions.

Execution Script Guidelines

Guidelines to write execution scripts include:

- All scripts require a result file. The result file must have a “1” in the first line to indicate success or “0” to indicate failure. The script can also include a descriptive text message for the failure, in the second line of this file. The client computer displays this text message.
- The execution script must contain at least one command-line parameter. This parameter is the name of the result file. By default, it is `$scriptresult$`. The client computer substitutes this file for the actual result file name at run time.
- The maximum size of an execution script is 10 MB.

You can create multiple execution scripts for an application. They can be designed to be invoked during the same or different phases of the application installation and execution. The Streaming Composer supports the following type of scripts:

- **Pre-Installation**—This script executes before the start of the virtual installation process.
- **Post-Installation**—This script executes after the virtual installation is complete, but before the execution of the application.
- **Pre-MSI Installation**—This script is only visible when you create an MSI package. It executes after the server downloads the MSI Installation files to the client computer, but before the MSI installation process starts.

The client computer executes the Pre- and Post-Installation scripts once for each virtual installation operation.

- **Pre-Execution**—This script executes before starting the application. So, this script runs each time any application runs.
- **Post-Execution**—This script executes after starting the application. So, this script runs each time any application runs.
- **Pre-UnInstallation**—This script executes before starting the uninstallation process.
- **Pre-UnInstallationRequirements**—This script executes before the Streaming Agent checks the uninstallation permission for the package. This script(s) performs actions that allow the package to be uninstalled (Example kill processes, stop services, unregister COM objects, close file handles, etc.).

This is required for certain packages such as Microsoft Office and Microsoft Project, which are not removed from the cache forever, due to other processes using the files from the cache of these packages. For instance, when the Microsoft Office process `ctfmon.exe` is running, this script would kill this application to facilitate easy removal of the package.

- **Post-UnInstallation**—This script executes after the un-installation of the application. The client computer executes the Pre-Installation and Post-Uninstallation scripts once for each uninstallation operation.
- **Data**—These files are required for other scripts. These files are not executed directly. Refer to *Appendix A : Execution Script-An Example* for a sample execution script.

To Add a Script

1. Navigate to **Package Information>Execution Scripts**.
2. Click **Add**. Select a script type from the drop down list and complete the optional parameters.
 - **Script Type**—Select the type of script from the drop-down list. The different types of scripts have been described above.
 - **Script**—Specify the script file location or use the browse button (...) to locate the script file.
 - **Parameters**—Specify the parameters that the package should pass to the script. Include the `$scriptresult$` parameter that specifies the name of the result file generated by this script.
 - **Description**—A free-text field for you to describe the purpose of the script. This information can be used by anyone who needs to rebuild the package.
 - **Error Message**—The error message to be displayed when the script fails. This can be a static string or alternatively taken from the result file, if specified.
 - **Bundle Script Executable with Package**—Select this option to bundle the script with the package. If you do not select this option, the Streaming Composer does not bundle the script with the package and the client computer assumes the presence of the script on the streamed system.

- **Run in Hooked Mode**—By default the external scripts are allowed to access the Global Registry only. The content of all streamed applications resides in the registry. Check this option to provide the external scripts a read and write access to the client computer registry.
- **Run with User Privileges**—By default scripts run under system privileges. A script that runs on a client computer desktop with system privileges cannot display a GUI, while an execution script that runs under user privileges can display a GUI. Check this option to allow the script to run under the current streamed user account. This option also needs to be used with scripts that need to access network resources.

! All packages that were created with special scripts for streaming services have to be repackaged without those scripts, as services are supported by default and no extra scripting is required to achieve the same.

Update the Application List

Package Information>Application displays a list of program executables that the Streaming Composer recognized in the application. You can specify additional executables, remove executables, or modify the attributes of currently defined executables. The Streaming Composer automatically identifies the applications that have a shortcut in the Windows Start menu.

To Add New Applications

You can specify additional executables which can also be available as shortcuts in the Windows Start menu.

1. Navigate to **Package Information>Application**. Update Application List is displayed.
2. Click **Add**. Enter details in the Add/Modify Application window using the following definitions to define an executable :
 - **Unique ID**—A unique identifier for the application, automatically provided by the Streaming Composer.
 - **Version**—An internal version number generated by the Streaming Composer, to indicate the version of the application.
 - **Name**—The name of the application or executable within the package. The application vendor usually supplies this name.
 - **Path**—The path to the executable file that launches the application.
 - **Parameters**—Command-line parameters required to launch the application.
 - **Working Directory**—The working directory for the application as defined in the path.

- **Vendor Version**—The version number assigned to the application by the vendor.
- **Shortcut Path**—The location of the Windows Start menu shortcut on the client computer. The Streaming Composer automatically detects shortcuts created under the Windows Start menu during the installation of the application. You can edit the Shortcut Path to customize the location of these shortcuts. Use the following variables to specify these locations:
 - `$userstartmenudir$` represents the Windows Start menu folder in the client computer system.
 - `$userprogrammenudir$` represents the Programs folder in the Windows Start menu on the client computer system.
 - `$userdesktopdir$` represents the desktop on the client computer system.

You can specify multiple shortcuts for this application by separating variables by a pipe character (“|”). For example, to create shortcuts on both the desktop and the Windows Start menu, specify the following shortcut path:

```
$userdesktopdir|$userstartmenudir$
```

- **Icon File**—The icon associated with the application on the Launch page and in the Windows Start menu. This icon can be in `.gif` or `.jpeg` format. The Streaming Composer can also convert icons from `.ico` to a `.gif` format. You can extract an icon from any executable. Streaming Composer extracts the icon and automatically creates a `.gif` image. To extract an icon, select an executable on the icon dialog. A second dialog displays a list of icons. You can select one from this list.

To Restrict Portal Page Shortcuts

Streaming Composer enables you to restrict the application icons displayed on the Launch page. You can choose to display only that application that you consider as the main application, to suit your requirements. Currently the user has the facility only to show all or one of the available applications; the user can not choose a set of applications and drop a set of applications.

1. Navigate to **Package Information>Application**. Update Application List is displayed.
2. Select the primary application that you wish to display on the Launch page. Click **Modify**.
3. Copy **Unique ID** on to the clipboard.
4. Navigate to **Package Information>Properties**. Update Package Properties is displayed.
5. Paste the copied value of Unique Id in **Installation Type** box after the statement `mainapp=.` For example: `mainapp=9dd227a6-fa39-44ba-b56a-e1e4b0642bd2.`

Generate Streamlets

Package Information>Streamlets section creates streamlets for the packaged application. Streamlets are the 4KB blocks that are transferred from the Streaming Server to the client computer. The Streaming Composer completes the values in the table once you create the streamlets. This step may take a few minutes, depending on the size of your application. To generate the streamlets, click **Run**.

Collect the Package Startup Block

Package Information>Startup Block creates a startup block for the packaged application. The startup block is a collection of streamlets required to start the application. The application does not start on the client computer desktop until the startup block has completed streaming.

The bigger the startup block, longer the time taken for the application to start up on first launch. The ideal startup block balances initial performance with application usability and functionality. Create multiple startup blocks until you find one that meets current and future user expectations. If the package contains multiple applications, consider creating a startup block with the most frequently used applications in the package.

Note: Reboot the system before you collect the startup block. This step clears any program or file caching in memory and results in a more accurate startup block.

To Create A Startup Block

1. Click **Collect**. In the dialog box that pops up, select the application executable from **Application** drop-down box. When you select an application, the Streaming Composer populates the remaining fields on the page. Click **OK**.

Note : You can also collect startup blocks for applications that do not have a `.exe` extension name but have secondary names like `.pdf`, `.cpl`, `.ppt`. Streaming Composer opens the file in the currently associated parent application.

2. The Streaming Composer immediately begins to collect the startup block. While the application runs, the Streaming Composer records the application file system usage to determine the minimal list of streamlets necessary to run the application. Do not close the application until you see a message that indicates that the Streaming Composer has finished collecting the startup block. If you did not finish running the application, you can continue to run it and stop the collection of the startup block manually.
3. When the Streaming Composer completes gathering information, it displays the number of blocks and streamlets it collected and the total size of the startup block.

Note : Click **Skip** to bypass collection of startup blocks now, and allow the Streaming Console to perform this activity once the package is deployed on the server. The best practice recommendation is to include the startup block in the package whenever possible.

Building the Package

Use this phase to build the package. The Streaming Composer collects the generated streamlets and the startup block and creates a zip file. To set an upper limit for the size of the zip file, set the value for the parameter `packager.build.zip.UptoLimit` in `packager.ini`. By default it is set to 2 GB. Any value other than 1 GB defaults to 2 GB. The name specified as Package Name in the Create New page, is assigned to this `.zip` file.

Please note that zip files cannot be created for packages with Unicode names. So while packaging such packages, set the flag `packager.build.zip.skip=true` in `packager.ini`. This setting forces Streaming Composer to skip the step to zip packages with Unicode names.

Streaming Composer creates log file —`<package name>.log`—in the package folder, for every package.

The complete package includes the following files and folders:

- **Package.txt**—Includes general information about the package.
- **Readme.txt**—Contains comments entered on the Package Properties page.
- **\Snapshot**—Contains pre-and post-installation snapshots.
- **\Intermediate**—Contains internal project data used by the Streaming Composer.
- **\Icons**—Contains the icons in this package.
- **ZIP file**—The archived file of the completed package. Upload this file to the Streaming Console.

To Build the Package

1. Navigate to **Build>Package**.
2. Click **Create**. The Streaming Composer builds the package. If the Streaming Composer finds orphan paths, it displays the Build Process Manager dialog box. Refer to the next section to resolve such orphan paths.
3. Click **OK** in the dialog box that pops up when the package is complete.
4. Click **Launch Streaming Console** to upload this package to the Streaming Console. To use this feature, you must specify the Control and Management URL in Tools> Preferences.

Note :

1. If the size exceeds 2 GB then Streaming Composer does not create a zip file. The package is available as a directory that can be uploaded to the Streaming Console.
2. For additional information on uploading a package, see the **Streaming System Administrator's Guide**.

Resolve Orphan Paths

An orphan path, does not exist in the package, but has path references in the registry, INI files, or in the dynamic files. You can create a rule to indicate, if the path reference should be pointed to a streamed file (inside the cache) or to an installed file (outside the cache). Make sure that the rule you create matches the application's behavior and expectations.

If orphan path references exist, then Build Process Manager dialog box pops up. It displays a list of orphan path references. Follow the steps given below to resolve these references.

1. Select the orphan reference and click **Add Rule**.
2. A new dialog box called Add/Modify Path Rule is displayed. The Streaming Composer displays all possible paths, including wildcard paths in a drop-down list. Select the appropriate path and select **Stream** or **Install**. A streamed file resides inside the cache. An installed file resides outside the cache.
3. Note that the rule appears in the lower window and the matching entries that comply with the rule, update their type column to indicate the correct action: Installed, or Streamed. The following choices are available:
 - **OK**—Saves your changes and closes the window. It accepts the new rules and builds the package.
 - **Abort**—Closes the window and aborts the build.
 - **Apply**—Saves your changes and creates the package. However, if the package encounters this or any other orphan paths, it stops the build and redisplay the Build Process Manager dialog box.
4. You can modify or remove a rule in the lower section. Click **Modify** to modify the rule - the Add/Modify Path Rule dialog box is opened again. Click **Remove** to discard the rule.
5. The Streaming Composer saves these rules in the Active Configuration Profile for later use.

Best Practices for a Snapshot Package

1. Start with a clean system that contains no other software components, other than those that are installed with the operating system. This system must have the same OS as the client computer target systems.
2. To ensure that the snapshot process captures all necessary files, use the lowest OS service pack on the system where you create the package.
3. Verify that your package supports the correct operating systems. Create a different package for each operating system.

4. Create a new user with local administrative privileges on the system where you create the package. Create unique user and computer names, that do not occur in the package. For example, a user name might be `streaming_user`. Login as this user.
5. Set up a Repository location on a drive different from the Windows drive, to store your packages.
6. Create two partitions on your packaging system. Place the package repository and the Streaming Composer on the secondary partition. Use the primary partition to create packages.
7. If you need to create multiple packages that require different settings, create a configuration profile for this group of packages.
8. The Streaming Composer automatically excludes unnecessary registry entries. However, it is recommended that you review the package contents and exclude file and registry entries that are not part of the application.
9. When packaging applications that have common components for sharing mode, it is recommended to make the common components installable for each application.
10. The Streaming Composer captures and lists all the packaged applications in the Windows Start menu. In some cases, there are additional applications that are located in the package, but are not exposed in the Start menu. You can add these applications during the packaging process.
11. You can document package information in **Package Information>Properties**. Use the free-text **Comments** field to document how you created the project, and add any custom modifications that you performed. The Streaming Composer saves this information to a `Readme.txt` file. This file is useful to those who need to update or modify the package at a later date.
12. Do not run applications or use external devices, such as a USB flash-drive, that may alter the state of the system during the packaging process. These applications and devices may interfere with the packaging process and lead to erroneous packages.
13. Collect the startup block for Snapshot packages during the packaging process. Reboot before you begin to collect the startup block.
14. Consider creating a startup block that contains the optimal amount of information, rather than the minimal amount. Include any additional application functionality that you think users may need.
15. Save the disk image before the second snapshot. It could be useful when you need to upgrade the package. Tools like Ghost or VMWare can be used.
16. Always install applications from within the Streaming Composer, especially if the application uses Windows Installer.
17. It is not recommended to have any antivirus turned on while creating Snapshot packages. Do not run Antivirus updates during this time.

18. It is recommended not to have any Windows updates or hotfixes running on the machine while creating Snapshot packages. Carry out these activities either after or before creating the package.
19. When .Net framework is required for a package, it is recommended that this framework be installed on the machine prior to the process of creating the Snapshot package.

Summary

The various steps involved in creating a Snapshot package has been described in detail in this chapter. The next chapter describes the steps involved in creating an MSI package.

Chapter 6

Creating an MSI Package

This chapter describes how to create a MSI package. The following sections describe the process in detail.

- *Creating a New MSI Package*
- *Defining the MSI Setup*
- *Reviewing the Content for an MSI Package*
- *Reviewing Information for an MSI Package*
- *Building the Package*
- *Best Practices for an MSI Package*

Creating a New MSI Package

Use the following procedure to create a MSI package:

1. Navigate to **Start>Programs>Altiris>Streaming Composer**. Execute the application.

Note : When you log on with a view to creating a new package, ensure that the user name and computer name are significantly different. If these are not different, then while creating a new package you see an error message - *To avoid packaging problems, user name and computer name should be significantly different*. Do not log on as Administrator while creating a package.

2. Click **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
3. Choose **MSI Package** in the Package Type section.
4. The **Package Path** is the folder path for the completed package. Enter the folder path or use the three-ellipses button to browse to the designated package folder. This path should not be the same as that of the path of the current operating system. It must be complete, as in the following example:

D:\myPackage1\PaintShopPro

5. The global configuration profile available is **Exclude_Setup_Info**. This configuration has the same values as the minimum configuration, except that it excludes any existing Windows Installer files and registry entries inside the package content.
(You can also choose any of the profiles already available in the system. Browse to the required folder and select the appropriate profile.)
6. Click **OK** and the Create Application screen is displayed, to take you through the steps of creating the MSI package.
7. The green arrow is positioned against **Import MSI** step in the Navigation Pane, to proceed further.

Defining the MSI Setup

To create a MSI package, as a first step you have to import the application MSI setup file(s) to the Streaming Composer. Define the location of the file(s) in the **MSI Path** and the **MSI Main File** fields:

- **MSI Path**—Defines the root location of the MSI setup. This folder should not have any other files other than the setup related. The contents of this folder and all sub-folders are included in the package.
- **MSI Main File**—Defines the main MSI setup file to be imported. It is automatically populated with the MSI setup files found in the selected **MSI Path** and its sub-folders. This folder should not have any other files other than the setup related files. The contents of this folder and all sub-folders are included in the package.

Streaming Composer also supports a self-extracting setup file. You only need to specify the self-extracting executable name and the folder where they are to be extracted.

To Define the MSI Setup

1. Click the **Import MSI** step. Import files and Create MSI Environment page is displayed.
 - a. **MSI Path**—Enter the MSI setup folder path or use the three-ellipses button to browse for the application MSI setup folder.
 - b. **MSI Main File**—Populated with the MSI setup files under the entered **MSI Path**. Select the main MSI setup file if more than one MSI file exists.
 - c. Click **Run**. The system processes the MSI setup files and converts the content and setup logic to a package structure.
4. You also have the option to import a self-extracting file. Click **Import Self-Extracting Setup**. Import Self-Extracting Setup Executable screen is displayed.
 - a. **Self-Extracting executable filename**—Enter the filename or use the three-ellipses button to browse for the filename.
 - b. **Output Folder**—Enter the folder in which the package would be stored. To browse for the folder click (...).
 - c. **Compress files**—Select to compress files.

- d. Click **OK** to import the files. It is recommended that you compress the files during this process.
5. Yet another method for defining the MSI Setup is the **Advanced** section. It gives you the option to perform MSI import and processing tasks on a one by one basis, facilitating interim customization.
 - a. Click the **Advanced** check box. The process options are enabled. To execute the task, highlight it and click **Execute Task**. The process options are:
 - Import MSI files
 - Extract MSI files
 - Process & Export MSI files
 - Import MSI files & Create Environment

Reviewing the Content for an MSI Package

In the next phase, package content is reviewed. It uses the manifest information, which is a read-only page. It contains a high-level summary of elements found in the MSI setup. All of these elements are not supported. The manifest information includes:

Manifest Element	Description
Windows File Protection	Windows file protection files in the package. This version does not support installation of WFP files.
Services	Services installed by the application. This version supports services like Windows Services etc. Some of the services and drivers like VPN client services are not yet supported by Streaming Composer.
Control Panel File	Control panel files in the package.
ODBC Driver	ODBC drivers in the package.
ODBC Data Source	ODBC data sources in the package.
Font	Fonts in the package.
File Association	File extensions associated with the package.
Startup Executable	Lists the executables that are launched on startup of the operating system. This version does not support startup executables.

Note : Streaming Composer supports packages which require installation of drivers. However it is essential that these drivers are not installed or streamed from within the cache folder. Hence, if a driver needs to be installed along with the streamed package, ensure that you specify (while packaging) that the driver be installed outside the cache.

Reviewing Information for an MSI Package

Use the Package Information Phase to update package properties, set system requirements, define execution scripts and applications, and split the application into streamlets.

Update (Define) Properties for an MSI Package

Use this section to specify additional package properties. The following table describes the fields on this page:

Package Property	Definition
Package Path	The location where the package is stored.
Original Package Path	The original package path of the upgrade.
Package ID	The unique identifier assigned by the Streaming Composer.
Source Path	A folder under package path in which empty folder structure of the application is created. This is enabled only if you select Create Source Folder .
Package Name	The Streaming Agent and the Streaming Console, use this to specify the package. You specified this name during the Create phase. You can change this name at any point during the packaging process.
Package Version	The package version number assigned by the Streaming Composer. The version number 1 applies to new packages. This number increments for each new upgrade version. The Streaming Console and the Streaming Agent, use this number to identify different versions.
Vendor Name	The name of the application developer or publisher.
Vendor Version	The version number assigned to the application by the vendor.
Installation Type	A free-text field. Use it to describe the type of media you used to install the application.
Comments	A free-text field. Use it to document any changes you made to the package during the build process. The information that you insert in this field becomes part of the build <code>readme.txt</code> file. This information is useful to anyone who needs to rebuild the package.

Review MSI Properties

Navigate to **Package Information>MSI Properties**. This is a read-only screen that describes the following MSI properties:

- **Product Code**—The unique identifier for the MSI setup file.
- **Required Engine Version**—The version of the Microsoft Windows Installer (MSI) engine required on the client computer desktop to execute the setup and to install the application package.
- **Primary Folder Property**—The MSI property name that defines the installation folder.

Set Up System Requirements for an MSI Package

Package Information>System Requirement displays a screen used to set the operating systems which support this package. This information is used by the Launch Server to open a Launch page that only has links to those provisioned applications that are supported by the operating system currently running on the client computer.

- **Requires reboot after installation**—Select the check box, if you want the client computer to reboot the system after installing the application.
- **Run with elevated privileges**—This option is selected, by default.

Define Execution Scripts for an MSI Package

Navigate to **Package Information>Execution Scripts** to define scripts that the client computer must execute before or after it installs, uninstalls, or executes the application.

An execution script can be any executable, such as a batch file. The client computer invokes the execution script with command-line parameters. You can create multiple execution scripts. The client computer invokes each script successively and examines the result after it completes the script. The application does not run if the execution script fails. Refer “*Define Execution Scripts*” on page 33 for more information.

Set Up Configurations for an MSI Package

To add or view or modify the primary MSI setup file, navigate to **Package Information>Setup configuration**. The screen displays the details of the primary MSI setup file. You can modify its attributes, specify a different setup executable, or specify a different deployment configuration.

- **Unique ID**—The unique ID assigned by the Streaming Composer.
- **Version**—The internal version assigned by the Streaming Composer to this package.

- **Name**—The name of the package provided by the application vendor and extracted from the MSI setup file.
- **Path**—The path of the MSI setup file, which can be an MSI file, an executable or a batch file.
- **Parameters**—Command-line parameters required to install the application. The Streaming Composer automatically determines the appropriate parameters, but you may override them.
- **Working Directory**—The working directory for the MSI setup as defined in the path.
- **Vendor Version**—The version number assigned to the application by the vendor.
- **Uninstall Parameters**—The parameters required to uninstall the application. For example, specify `/qn` to uninstall the application in silent mode.
- **Icon File**—The icon associated with the application on the Launch page and in the Windows Start menu. You can extract an icon from any executable. Streaming Composer extracts the icon and automatically creates a `.gif` image. The icon may also be in a `.jpeg` format. To extract an icon, select an executable on the icon dialog. A second dialog displays a list of icons.

To Add or Modify a Setup Configuration

1. On the Setup Configuration page, click **Add** or **Modify**.
2. The Add/Modify setup configuration is displayed and enter the parameters.
3. Click **OK** to save the entries.

By default, the MSI file is displayed. However, if the `setup.exe` is part of the installation file, then use it. When you choose a `setup.exe` file, the Streaming Composer automatically determines the appropriate command-line parameters.

While packaging MSOffice XP Professional, if you have created a transform file for office XP application, then these transforms can be applied to this package.

1. On the Setup Configuration page, click **Modify**.
2. In the Add/Modify setup Configuration screen displayed, Edit Parameter. Add `TRANSFORMS="OfficeXP.MST"` at the end of the current text (i.e., after the `ALLUSERS=2`).

Note : If this `.mst` file has been saved in a folder other than installation location, then specify the absolute path of the `.mst` file. Example :

```
TRANSFORMS="C:\OfficeXP\OfficeXP.MST"
```

Generate Streamlets for an MSI Package

The Generate Package Streamlets page opens. The Streaming Composer has completed the interim steps between Import MSI and Generate Streamlets. In this step, the Streaming Composer creates streamlets for the MSI application. Streamlets are the 4 KB blocks that are transferred from the Streaming Server to the Streaming Agent. The Streaming Com-

poser completes the values in the table, once you create the streamlets. This step may take a few minutes, depending on the size of the application. To generate the streamlets, click **Run**.

Building the Package

In this phase the package is actually built. Navigate to **Build>Package**. In Create Final Package for Streaming click **Create** to create the package zip file. The Streaming Composer collects the generated streamlets while it creates the zip file. To set an upper limit for the size of the zip file, set the value for the parameter `packager.build.zip.UptoLimit` in `packager.ini`. By default it is set to 2 GB. Any value other than 1 GB defaults to 2 GB. Subsequently this zip file is uploaded on to a Streamlet Engine by the Streaming Console. Refer to the **Streaming System Administrator's Guide** for more information on how to upload packages.

Note : If the size exceeds 2 GB then Streaming Composer does not create a zip file. The package is available as a directory that can be uploaded to the Streaming Console.

Streaming Composer creates log file —`<package name>.log`— in the package folder, for every package.

If you wish to upload this package to the Streaming Console using Streaming Composer, click **Launch Streaming Console**. To use this feature, you must define the URL for the Streaming Console in Tools> Preferences. Refer to the section on “*Defining the Preferences*” on page 23, for details.

The complete package project includes the following files and folders:

- **Package.txt**—Includes general information about the package.
- **Package.pdr**—Includes information about the package creation like vendor name, Streaming Composer version used etc.
- **Readme.txt**—Contains comments entered on the Package Properties page.
- **\MSI-SRC**—A backup of the original MSI setup files.
- **\MSI-DEST**—Contains internal project data used by the Streaming Composer.
- **\Intermediate**—Contains internal project data used by the Streaming Composer.
- **\Icons**— Contains a list of the icons in this package.
- **ZIP file**—The archived file of the completed package. Upload this file to the Streaming Console.

Install a Specific Application

The procedure explained in the previous sections install all the applications on the client computer, by default. This section explains the work around to install only one particular application from the MSI file.

By default, the client computer adds `ADDLOCAL=ALL` to the MSI installation to install all applications from the MSI file. To just install any one of the executables in a MSI file:

1. Add the following parameter to the package `.pdr.xml` file (in the package folder) manually :

```
<PROPERTY NAME="MSI:DisableForceAllLocalInstall" VALUE="1" />
```

2. Open the MSI package in Streaming Composer.
3. Navigate **Build>Package** and recreate the package.

Best Practices for an MSI Package

1. If more than one MSI file exists for the application, you must identify the main MSI file for the package.
2. Place the MSI setup files in a separate folder to ensure that the package includes only necessary files.
3. The original MSI installation must work in silent mode and accept `InstallDir` or `TargetDir` to identify the target installation directory. The Streaming System uses these MSI properties to direct the application files to the Streaming Agent cache.
4. A single MSI installation provides better streaming than several smaller combined MSI installations, that activate each other.
5. It is recommended that if the MSI installation requires that the system reboot, the installation must install all files before it reboots the system.
6. Set up a Repository location to store the packages created with the Streaming Composer.
7. While packaging applications that have common components for sharing mode, it is recommended to make the common components installable for each application.

Summary

This chapter has described the various steps involved in creating an MSI package. The next chapter describes the process of creating a Tracking Package that can monitor the usage of conventionally installed applications.

Chapter 7

Creating a Tracking Package

Tracking Packages enable monitoring the usage of conventionally installed applications. Each Tracking Package is configured to track a set of predetermined applications. A Tracking Package is associated with specific application(s).

This chapter describes how to create a Tracking package. The following sections describes the procedure in detail.

- *Create Phase*
- *Package Building Phase*

Create Phase

To create a Tracking Package, set the package type (of a new package) as Tracking and associate the application(s) to be tracked using this package.

It is essential that the application(s) to be tracked (using this Tracking Package) be installed on the machine being used for creating the Tracking Package.

1. Navigate to the Streaming Composer on the Windows Start menu: **Start>Programs>Altiris>Streaming Composer**. Execute the application.
2. Click the **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
3. On the Create New page displayed, choose the **Tracking Package** option.
Enter the folder path or browse to the designated package folder. It must be a full path and cannot be a network path. For example, `D:\myPackages\PaintShopPro`.
4. The **Package Name** is the name you assign to the package. Enter a name. It is strongly recommended that the name describes the application being monitored by this package.

! Do not include special characters like “&” in the package name.

5. By default, the configuration profile is selected as **Default**. To learn more about configuration profiles, refer to the section, “*Using Configuration Profiles*” on page 17.
6. Click **OK** and the Create Tracking Package screen is displayed, to take you through the steps of creating the tracking package. As a next step, applications to be tracked as

a part of this package, need to be listed. Click **Tracking Applications List** in the Navigation Pane.

Applications

Each tracking package is designed to track one or more applications that constitute a package. **Create Tracking Package>Applications** allows you to specify these applications.

In Tracking Applications List, the following parameters of the applications included in this Tracking Package are displayed :

- **Name**—The name of the application executable to be tracked.
- **Add**—Click to add an application to the existing list (if any) of applications, to be tracked by this package.
- **Modify**—Click to modify the parameters of the selected application.
- **Remove**—Click to remove the selected application from this list.

Add Application to Tracking Application List

1. Click **Add**. Tracked Application page is displayed.
2. Enter the **Name** or browse for the application to be included. Click **OK**. Tracking Applications List is displayed again.
3. Repeat the above steps to include the required applications.

Application Identification

Create Tracking Package>Application Identification enables you to define rules that help Streaming Composer discover the tracked applications based on the product name. Streaming Composer displays the name of the programs as displayed in the Add/Remove Programs in Control Panel. Select the applications to be included in this Tracking Package.

This is not a mandatory step. When a Tracking Rule is not specified, then the application is identified only when it is executed. When a Tracking Rule is defined and product-id is available then, license is consumed when Tracking Package is enabled on the client computer. In the absence of the product-id, the license is consumed only when the application is executed.

In Define Tracking Rule the following parameters are displayed :

- **Rule Name**—A rule name that helps identify the applications by their product name.
- **Rule Value**— A list of product names as displayed in the Add/Remove Programs in Control Panel. Select the application to be tracked.
- **Add**—Click to add a new rule for an application.

- **Modify**—Click to modify an existing rule.
- **Remove**—Click to delete an existing rule.

To Add a New Tracking Rule

1. Click **Add**. Add Tracking Rule is displayed.
2. Select the product name from the list displayed in **Application Identification**. Click **OK**. Define Tracking Rule becomes active.

System Requirement

Create Tracking Package>System Requirements lists the operating systems that support the package. By default, the required operating systems are automatically selected as shown below.

Define Execution Scripts

Create Tracking Package>Execution Scripts is used to define the scripts that the client executes (with command-line parameters) for some milestones like pre-uninstallation for an application. On completion of the execution, the result is examined. Click **Add** to open Define Execution Scripts screen.

To ensure that a conventionally installed package is always monitored, you must include a pre-uninstallation script for this application. When the validity period of the application (set using the Streaming Console) ends, you can choose the course of action to be followed. Use the client configuration page in the Streaming Console to specify your choice. On this page, if you select **Remove Application only**, then on expiry of the validity period, the conventionally installed application is removed from the cache, by this script.

Sample pre-uninstallation scripts are available in `\Program Files\Altiris\Streaming Composer\Sample\Tracking Package`. Refer “*Script to uninstall any MSI installation silently*” on page 81 for details. Refer “*Execution Script Guidelines*” on page 33 for guidelines to write execution scripts.

To Add a Script

1. Navigate to **Package Information>Execution Scripts**.
2. Click **Add**. Select a script type from the drop down list and complete the optional parameters.
 - **Script Type**—Select the type of script from the drop-down list. Pre-uninstallation and post-uninstallation scripts need to be created for a Tracking Package.

- **Script**—Specify the script file location or use the browse button (...) to locate the script file.

! Installshield applications require user interaction for uninstallation. To facilitate the activation of uninstallation GUI, select **RUN WITH USER PRIVILEGE (non system account)**, while setting parameters for pre-uninstallation script.

For more information on the other parameters refer “*To Add a Script*” on page 34.

Package Building Phase

Use this phase to build the package. To Build the Package :

1. Navigate to **Build>Package**.
2. Click **Create**. The Streaming Composer builds the package.
3. Click **OK** in the dialog box that pops up when the package is complete.
4. Click **Launch Streaming Console** to upload this package to the Streaming Console. To use this feature, you must specify the Control and Management URL in Tools> Preferences.

For more information refer “*Building the Package*” on page 38.

Summary

The various steps involved in creating a Tracking package has been described in detail in this chapter. The next chapter describes the steps involved in creating a SVS package using .VSA files.

Chapter 8

Creating a Package for SVS Applications using a .VSA file

This chapter describes how to enable Software Virtualization Solution (SVS) packages for streaming. SVS is an application virtualization solution from Altiris. SVS enables abstraction of installation files of an application, so that the application can be installed without altering the operating system settings. Hence the application can be installed without conflicting with other installations.

The Virtual Software Archive (.VSA) file, created using SVS, can be converted to a streaming package using Streaming Composer. The resulting file can then be streamed. All the advantages of provisioning, license management and reporting still apply to these virtualized applications.

Please refer to SVS documentation for details on creating .VSA files. SVS package can be created using the Streaming Composer GUI or the command-line. Refer “*Creating SVS Packages*” on page 77 for more information on the use of command-line for this purpose.

The following sections describes in detail, the procedure for creating a package using the GUI.

- *The Create Phase*
- *Building the Package*

The Create Phase

Use the following procedure to create a SVS package:

1. Navigate to **Start>Programs>Altiris>Streaming Composer**. Execute the application.
2. Click **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
3. Choose **SVS Package** in the Package Type section.
4. The **Package Path** is the folder path for the completed package. Enter the folder path or use the three-ellipses button to browse to the designated package folder. It must be complete, as in the following example:

```
D:\myPack1\PShopPro.
```

Note : Very long names for .vsa files result in failure of packaging process. So it is recommended that SVS packages be created with short package path names.

5. Select Configuration Profile as **Default**.
6. Click **OK**.
7. This is followed by the **Import VSA** step in the Navigation Pane. Enter the file name or use the three-ellipses button to browse to the required folder to specify the name of the .vsa file.
8. Click **Import**. The progress of the import process is displayed and the control moves to the next step.

The Package Information Phase

All the information about system requirements, setup configuration etc. is automatically extracted from the .vsa file. The package name is also read from the .vsa file and used as the package name.

For more information on this phase, refer “*The Package Information Phase*” on page 31. This carries information on Properties, System Requirements, Execution Scripts and Updation of Application List.

Building the Package

The steps to build the package are :

1. Navigate to **Build>Package**.
2. Click **Create**. The Streaming Composer builds the package.
3. Click **OK** in the dialog box that pops up when the package is complete.
4. Click **Launch Streaming Console** to upload this package to the Streaming Console. To use this feature, you must specify the Control and Management URL in Tools> Preferences.

For more information refer “*Building the Package*” on page 38.

Upgrading SVS Package

Streaming Composer allows you to upgrade a SVS package independent of the system image used for the original package. You can create an upgrade using any machine and any login. The prerequisites for this are:

- The original package should be a SVS package.
- The upgrade package should be a .vsa file created using SVS.

To upgrade a SVS package:

1. Navigate to **Start>Programs>Altiris>Streaming Composer**. Execute the application.
2. Click **Create New tab** in the New/Open Package dialog box or select **File>New** option from the menu.
3. Select **SVS package** from the Package List.
4. Click **Upgrade Package** check box.
5. Browse or enter the **Last version package path**.
6. Browse or enter the upgrade package path.
7. Click **OK**.

Now, create the package as described in section 8.1. The Streaming Composer proceeds to the next step.

Summary

The various steps involved in creating a SVS package has been described in detail in this chapter. The next chapter describes the steps involved in upgrading a package.

Chapter 9

Upgrading a Package

When applications go through a version change, upgrade-packages are made available by the vendor, to enable upgradation of the existing version. You can upgrade the original package using the upgrade-package.

An upgrade-package has the same name and package ID as the base version. The Streaming Composer examines the difference between the original application and the upgrade and creates the upgraded version. If the client has the package installed, then the server streams the upgraded version. When the client installs the application for the first time, the server installs the newer version. The Upgrade feature is useful for creating upgrade packages successively across versions.

The upgradation process explained in this section is applicable to Streaming Composer versions 4.0 and above. To upgrade packages created using earlier versions, first convert the package to adhere to the version 5.2.1 format and then follow the process detailed below.

Ensure that you take a backup of the existing package before you start the upgrading process. To take a backup, use the **Tools>Backup** option in the menu bar.

Machine Independent Packaging

You can upgrade Snapshot packages independent of the machine used to create the package. You do not have to save original system images using software such as Ghost or VMWare, to upgrade packages of existing Snapshot packages.

You can create upgrade packages on any machine and with any user login id, i.e. create the package for version 1 on machine 'A' with user login 'A' and then create the upgrade package on machine 'B' with user login 'B'. The restrictions that apply to this feature are:

1. The operating system and service pack should be the same on both the machines.
2. The previous version of the application must be installed on the machine where the upgrade package is to be created.

Example using Office 2003

1. Log on to Machine 'A' with user name 'A'.
2. Create a Snapshot package of Office 2003 on Machine 'A'.

3. Move to Machine 'B' which has the same operating system, service pack and same version of IE and login id as user 'B'.
4. Install Office 2003 conventionally.
5. Then create an upgrade package for Office 2003 SP1 on machine 'B'.

Note : Usually the upgrade package is created in the same root drive as that of the original package. If the root drive in the new machine (used to create the upgrade package) is different, then use the **Tools>Monitoring>Files** option to change the root drive to be used. Do this before snapshots are taken. This ensures that the upgrade package is created without a problem even though the root drive is different from that of the original package.

Upgrading a Snapshot Package

Streaming Composer allows you to upgrade a Snapshot package independent of the system image used for the original package. You can create an upgrade using any machine and any login id. The prerequisites for this are:

- The operating system and service pack should be the same as that of the original system.
- The previous version of the application must be installed on the machine where the upgrade package is to be created. The installation could be done using conventional methods.

! All packages that were created with special scripts for streaming services, have to be repackaged without those scripts, as services are now supported by default and no extra scripting is required to achieve the same.

Note : If the original installation disk or the PC is not available, then use the *Conventional Package Installation* feature to first install the package and then continue with the upgrade process.

1. Navigate to **Start>Programs>Altiris>Streaming Composer**. Execute the application.
2. Click **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
3. Select **Snapshot package** from the Package List.
4. Select **Upgrade Package** check box.
5. Browse or enter the **Last version package path**.
6. Browse or enter the **Package Path**.
7. Click **OK**.

Now, create the package as described in Chapter 5. The Streaming Composer proceeds to the next step.

Note : Usually the upgrade package is created in the same root drive as that of the original package. If the root drive in the new machine (used to create the upgrade package) is different, then use the **Tools>Monitoring>Files** option to change the root drive to be used. Do this before snapshots are taken. This ensures that the upgrade package is created without a problem, even though the root drive is different from that of the original package.

Note : To upgrade using command-line interface, refer to “*Upgrade Snapshot Package*” on page 73.

Conventional Package Installation

This feature allows the system administrator to install or restore an existing Snapshot package (before upgrading this package) without using the original installation disk.

1. Navigate to **File >Open Existing** to open an existing Snapshot Package. This step is essential as Install Package option is enabled only when an existing package is opened.
2. Navigate to **Tools>Install Package**.
3. The Install Package dialog box is displayed. Select the **Package** tab to see the package information.
4. By default all the components of the package i.e. files, registry, shortcuts and scripts are installed in the system. To install selectively, select the **Advanced** tab and select the required options.
5. Click **Install** to install the package. Click **Cancel** to stop the process.
6. A progress bar shows the status of the current installation. The time taken for installation depends on the package size. Once the package is successfully installed a popup is displayed. Click **OK** to continue.

Note : Some packages require that the system be restarted on completion of the installation process. To restart the system now, click **Yes** in the popup dialog box that appears on successful completion of installation. Click **No** to restart later.

This feature does not include package un-installation. To undo the package installation, use other disk backup/restore software or VMWare.

Upgrading an MSI Package

Verify that the upgrade package has the same MSI product code as the base MSI application.

! All packages that were created with special scripts for streaming services, have to be repackaged without those scripts, as services are now supported by default and no extra scripting is required to achieve the same.

1. Navigate to **Start>Programs>Altiris>Streaming Composer**. Execute the application.
2. Click **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
3. Select **MSI Package** from the Package Type list.
4. Select **Upgrade Package** check box.
5. Browse or enter the **Last version package path**.
6. Browse or enter the **Package Path**.
7. Click **OK**.

Now, create the package as described in Chapter 6. The Streaming Composer proceeds to the next step.

Note : To upgrade using command-line interface, refer to “*Upgrade Snapshot Package*” on page 73.

Creating an MSI Patch Package

MSI Patch is a special type of Windows Installer file that patches existing installations. An MSI Patch package can be created by Streaming Composer for these types of files. Like MSI upgrade packages, MSI Patch packages are based on base MSI packages. Create a package with multiple patches by creating successive patch packages based on the previous package.

For example, consider a case where two patches `Patch1` and `Patch2` are available for the original MSI package named `BaseMSI`. First, create a patch package `PackagePatch1` for `Patch1`. This package contains the files of `Patch1` and all the contents of `BaseMSI`, the original MSI package. A patch package `PackagePatch2` is now created for `Patch2` based on the `PackagePatch1`. The package `PackagePatch2` has all the files of `Patch2` in addition to `Patch1` files and original MSI package files. Use this method to create patch packages.

The client checks for patches on the desktop that have not been applied and then applies each patch successively. If the client detects that none of the patches have been applied, it applies `patch1` first and then `patch2`. If it detects that `Patch1` has already been applied, it applies `Patch2` only.

You can also upgrade an MSI Patch package. Use the Patch package as a base package to create the upgrade package.

! Ensure that the patch `.MSP` file is not in the same folder as the `.MSI` file, used to create the original package.

To Create an MSI Patch Package

1. Navigate to **Start>Programs>Altiris>Streaming Composer**. Execute the application.
2. Click **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
3. Select **MSI Package** from the Package Type list.
4. Select **Upgrade Package** and **MSI Patch Package** check boxes.
5. Browse or enter the **Last version package path**.
6. Browse or enter the **Package Path**.
7. Streaming Composer automatically takes you to first step of creating a MSI package.
8. Create a new MSI Package.
9. In the MSI setup window, enter the path of the `.MSP` file. You can either define the MSI path or perform MSI import, and the processing tasks one by one to customize any interim stage.
10. The Set Configuration screen displays the MSI Patches contained in the package. You can modify the following attributes or specify a different MSI Patch file for another deployment configuration.
 - **Unique ID**—The unique ID assigned by the Streaming Composer.
 - **Name**—The name of the patch. The Streaming Composer assigns a name that can be changed. The name is based on the patch number. For example, the possible names are `Patch_1` or `Patch_2`.
 - **Path**—The path of the MSI Patch file (an MSI patch file, an executable or a batch file).
 - **Parameters**—Any command-line parameters required to install the application. The Streaming Composer automatically determines the appropriate parameters.
 - **Working Directory**—The working directory for the MSI setup as defined in the path.
 - **Vendor Version**—The version number assigned to the application by the vendor.
 - **Version**—An internal version number assigned.
11. Generate package streamlets.

Note :

1. This version of Streaming Composer does not support Operating System patches. So, applications that are modified by an Operating System patch, cannot be upgraded by the method described in this section.
2. If the `.MSP` file is not available, then files from the `setup.exe/update.exe` must be extracted manually.

Summary

This chapter has described the steps involved in upgrading a package, whenever an upgrade is made available for a packaged application.

The next chapter discusses the nuances of the command-line interface.

Chapter 10

Using Command-Line for Packaging

Streaming Composer includes a special Command-Line Interface feature that enables you to create new packages and upgrade packages using the command-line. This feature is implemented for both Snapshot packages and MSI packages. Files/shortcuts and execution scripts (contained in Packages) and additional package properties can also be updated.

While packaging large applications which have a large number of files and folders and very few or no registry entries, significant user interface is required to complete the packaging process. This is also a time consuming process. Such applications are ideally suited for packaging using the command-line interface.

This chapter describes the process of package creation and upgradation of packages using command-line interface. The following sections describes the procedure in detail.

- *Information on files and paths*
- *Create New Snapshot Package*
- *Upgrade Snapshot Package*
- *Creating MSI Packages*
- *Upgrade MSI Package*
- *Creating SVS Packages*

Information on files and paths

A very essential pre-requisite for using the command-line interface for the packaging process is to define and create a directory structure on the local machine. This is called the Source File Path. It is also essential to create `.xml` file which holds the package description properties. This is one of the arguments passed to the command-line.

This section describes the hierarchy of the path and the role of the subdirectories in this path. The contents of the `.xml` file and their implications have also been explained. Some additional tips (including shortcuts to the application) and additional package customizations have also been included.

Source files path content

The sources files path is a directory structure to be created on the local machine. Streaming Composer places the files of the application in the appropriate folders according to their nature. It is hence essential that this directory structure (of source files path) be exactly the same as the structure to be streamed (including files/shortcuts) on the client computer.

In the following example, “N:\Org\customer1\ Files\” is the source files path. The files to be packaged include:

- files to be streamed
- shared system files to be installed to [current Window Drive:]\Windows\system32 directory
- files to be installed in [current Window Drive:]\Windows

Streaming Composer offers a list of environment variables that can be used to define such paths. The directory names inside source files path should be Streaming System environment variables. This allows the administrator to deploy files to system and shared folders. For example :

- \$APPS_PATH\$—to hold the files to be streamed to the cache
- \$windrive\$—to hold files to be installed in [current Window Drive:]\Windows
- \$systemdir\$—to hold files to be installed in [current Window Drive:]\Windows\system32 directory

The directory and files under \$APPS_PATH\$ are streamed. All other folders are considered as “installable”.

Hence, include these subdirectories in the source file path created on the machine. This implies that,

- N:\Org\customer1\ Files\ \$APPS_PATH\$ holds the files to be streamed.
- N:\Org\customer1\ Files\ \$systemdir\$ holds the shared system files to be installed in [current Windows Drive:]\Windows\system32 directory.
- N:\Org\customer1\Files\ \$windrive\$ holds the files to be installed in [current Windows Drive:]\Windows

To install other packages using the same executable, use additional folders like:

```
N:\Org\customer2\ Files\.
```

The executable information should not contain any Streaming System environment variables. This enables Streaming Composer to extract the icon and executable information and automatically create the appropriate shortcut.

Note : The source file is created only by Streaming Composer 4.5 and above. Hence, to upgrade a package (created on an older version) using the command-line, re-create it using the newer version. Only the last step i.e. “Create package” needs to be re-run.

Refer to *Sample File Structure* for examples of Source files path. Refer to *Environment variables mapping list* for a list of Streaming System environment variables to be used in the directory structure.

XML File

The `.xml` file contains package description properties that are used while creating the package:

1. **Update:SourceFilePath**—Reference to the source files root directory. This directory contains the files that are to be a part of the package. It should not be shared with other packages.
2. **Update:ConfigSet**—Configuration profile to be used on initial creation of a package (Default, Exclude_Setup_Info, etc.).
3. **Update:ExternalExecutablePath**— External executable to be run before finalizing the package (before collection of blocks, and zip file creation) in order to externally update the package xml files.
4. **Update:ExternalCommandLineParams**—command-line parameters to be used while running the external executable. Use `$packagedir$` in order to replace (at runtime) with the package path location.

These are mandatory properties required by Streaming Composer, hence should be defined in the `.xml` file. Additionally, some package properties can also be changed, to allow some flexibility. These properties include the package name, package version etc. For example:

```
<PROPERTY NAME="PackageName" VALUE="ProcExp test"/>
<PROPERTY NAME="PackageVersion" VALUE="2"/>
```

Refer to *List of supported properties* for a list of property names

Generate an `.xml` file from an existing `.pdr.xml` file:

This section helps you to create the `.xml` file using the `.pdr.xml` file provided by Streaming Composer for every application that has been installed (using Streaming Composer). `.pdr.xml` file is available in the folder where `package.zip` has been created.

1. Create a template `.xml` file using the example given below.
2. Create a temporary Snapshot package using the Streaming Composer GUI.
3. Open the `*.pdr.xml` file created by this package.
4. Copy/paste the contents of the template `.xml` file created in step 1 to the `.pdr.xml` file and save the file as the `.xml` file.

Note : Save this file as `input.xml` when you create a package and save it as `update.xml` when you upgrade an existing package. Please also ensure that the section `Description` is changed to `DescriptionUpdate` before saving the file, as given in the example below:

Example of .xml file content

```
<PACKAGE>

  <DESCRIPTIONUPDATE>

    <PROPERTY NAME="Update:SourceFilesPath"
VALUE="N:\folder1\customer1\files"/>

    <PROPERTY NAME="Update:ConfigSet" VALUE="default"/>

    <PROPERTY NAME="Update:ExternalExecutablePath"
VALUE="N:\folder1\prog1.bat"/>

    <PROPERTY NAME="Update:ExternalCommandLineParams" VALUE="$package-
dir$"/>

  </DESCRIPTIONUPDATE>

</PACKAGE>
```

ShortCuts

Applications to be shown on the Launch page require a shortcut to be defined in the source files path. To define this shortcut, first create a shortcut in the directory where the application resides. Then copy it to the relevant folder. For example, copy the shortcut to `$user-programmenudir$`, to show the shortcut in the Program menu under the Start menu, or copy it to `$desktopdir$` to show it on the desktop.

Additional Customization

An external executable or batch file can be defined in the `.xml` file. You can use this feature for additional package customization, before finalizing the package. This `.exe` is run before collecting the package blocks and creation of the zip file. This batch file is specified for the property : `Update:ExternalExecutablePath` defined in the `.xml` file.

Note : Carry out these changes cautiously, since there are very few validity checks. Ensure that the external file does not modify any other file. It is very important that the meta `.xml` file schema is fully understood before making any changes.

The external process can update only the following files:

1. ***.pdr.xml**—This is the package description file and contains the package properties, shortcuts and scripts information.
2. **Intermediate\MetaAppInfoFile.xml**—The package files metainfo file.
3. **Intermediate\MetaAppInfoReg.xml**—The package registry metainfo file.

4. **Intermediate\MetaAppInfoINI.xml**—The package INI metainfo file.

Samples

Sample File Structure

In the following example, the source files path used while creating the package is "N:\Org\customer1\ Files\". The other specifications are:

- files to be streamed—`sfile1, sfile2`
- shortcut to application under Program menu—`app1.lnk`

The directory structure of the source files path is given below. The levels of indentation represents the directory hierarchy.

```
N:\Org\customer1\ Files\
    $APPS_PATH$\ // files stored in the cache
        sFile1
        sFile2
    $programmenudir$\ // all-users menu directory
        app1.lnk
    $systemdir$\ // system32 directory (optional)
        File3.dll
```

Source File Path for Package Upgrade

You need to define source files path for the upgrading a package. This definition is in addition to that used for creating the package.

In this example, the source files path used while upgrading the package is

```
N:\Org\customer1\upgradeFiles\.
```

- files to be streamed—`upfile1, upfile2`
- shortcut to application under Program menu—`up_app3.lnk`

The directory structure of the source files path is given below. The levels of indentation represents the directory hierarchy.

```
N:\Org\customer1\ upgradeFiles\
    $APPS_PATH$\ // files stored in the cache
        upFile1.exe
```

```

        upfile2.chm
$programmenudir$ \ // all-users menu directory
        up_app3.lnk

```

Streaming System environment variables mapping list

```

$APPS_PATH$ ==> Streaming Agent cache (c:\AltirisCache\fltroot\xxxx\)
$systemdir$ ==> C:\WINDOWS\SYSTEM32
$fontsdir$ ==> C:\WINDOWS\FONTS
$userappdatadir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\APPLICATIONDATA
$userfavortiesdir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\FAVORITES
$userdesktopdir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\DESKTOP
$usermypicturesdir$ ==> C:\DOCUMENTS AND SETTINGS\ASUser\MY DOCUMENTS\MY
PICTURES
$userpersonaldir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\MYDOCUMENTS
$usersenttmdir$ ==> C:\DOCUMENTS AND SETTINGS\ASUser\SENDTO
$useradmintoolsdir$ ==> C:\DOCUMENTS AND SETTINGS\ASUser\START MENU\PRO-
GRAMS\ADMINISTRATIVE TOOLS
$userstartupdir$ ==> C:\DOCUMENTS AND SETTINGS\ASUser\START MENU\PRO-
GRAMS\STARTUP
$userprogrammenudir$ ==> C:\DOCUMENTS AND SETTINGS\ASUser\START
MENU\PROGRAMS
$userstartmenudir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\STARTMENU
$usertemplatedir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\TEMPLATES
$userlocalappdata$ ==> C:\DOCUMENTS AND SETTINGS\ASUser\LOCAL SET-
TINGS\APPLICATION DATA
$tempdir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\LOCALSETTINGS\TEMP
$userlocalsettings$ ==> C:\DOCUMENTSANDSETTINGS\ASUser\LOCALSETTINGS
$userprofiledir$ ==> C:\DOCUMENTSANDSETTINGS\ASUser
$desktopdir$ ==> C:\DOCUMENTSANDSETTINGS\ALLUSERS\DESKTOP
$appdatadir$ ==> C:\DOCUMENTSANDSETTINGS\ALLUSERS\APPLICATIONDATA
$admintoolsdir$ ==> C:\DOCUMENTS AND SETTINGS\ALL USERS\START MENU\PRO-
GRAMS\ADMINISTRATIVE TOOLS
$startupdir$ ==> C:\DOCUMENTS AND SETTINGS\ALL USERS\START MENU\PRO-
GRAMS\STARTUP

```

```

$programmendir$ ==> C:\DOCUMENTS AND SETTINGS\ALL USERS\START MENU\PRO-
GRAMS

$startmenudir$==>C:\DOCUMENTSANDSETTINGS\ALLUSERS\STARTMENU

$templatedir$==>C:\DOCUMENTSANDSETTINGS\ALLUSERS\TEMPLATES

$profiledir$==>C:\DOCUMENTSANDSETTINGS\ALLUSERS

$commonprogramfilesdir$==>C:\PROGRAMFILES\COMMONFILES

$programfilesdir$==>C:\PROGRAMFILES

$windir$ ==> C:\WINDOWS

$windrive$ ==> C:\

```

List of supported properties

The list of valid properties is given below.

- PackageName
- PackageVersion
- OSPlatforms
- VendorName
- VendorVersion
- ApplicationName.x
- ApplicationExecutablePath.x
- ApplicationCommandLineParams.x
- ApplicationWorkingDirectory.x
- ApplicationVendorVersion.x
- ApplicationShortcutPath.x

The following section is an extract from .xml file that includes some definitions that change the corresponding package properties:

```

<PROPERTY NAME="PackageName" VALUE="mypackage" />
<PROPERTY NAME="PackageVersion" VALUE="1" />
<PROPERTY NAME="OsPlatforms" VALUE="Windows NT 5.1" />
<PROPERTY NAME="PreInstallScriptType.1" VALUE="Data" />
<PROPERTY NAME="PreInstallScriptExecutablePath.1" VALUE="C:\win-
help\prog1\PROG1.HLP" />
<PROPERTY NAME="PreInstallScriptCommandLineParams.1" VALUE="" />

```

```

    <PROPERTY NAME="PreInstallScriptDescriptionMessage.1" VALUE="*EXECUT-
ABLE*" />

    <PROPERTY NAME="PreInstallScriptErrorMessage.1" VALUE="*EXECUT-
ABLE*" />

    <PROPERTY NAME="PreInstallScriptBundleWithPackage.1" VALUE="1" />

    <PROPERTY NAME="PreInstallScriptAttributes.1" VALUE="" />

```

Create New Snapshot Package

Command Line Syntax

Syntax:

```
Packager.exe /updatexml:<xml file> <package-path>
```

where

/updatexml:<xml file>—Create new package using xml file

<package-path>—The location where the package file is created

The Streaming Composer exits with the following error codes:

0 - Success

1 - Failure (check logs)

2 - Package is invalid due to empty files content or to indicate that no changes have been found to create an upgrade package.

As discussed above, the input .xml file provides the data required by the Streaming Composer to create packages and is maintained by the Administrator and customized as needed for each package. It contains initial package description, properties, configuration profile, and the location of the source files.

Example:

```
Packager.exe /updatexml:n:\Org\customer1\customer1.xml
n:\Org\customer1\package
```

Streaming Composer creates a new package in the location n:\Org\customer1\package using the information in the input .xml file—n:\Org\customer1\customer1.xml.

Procedure

1. Create a directory tree structure as described in Source Files Path and the set root of this tree as the default “Source Root Folder”.
2. Place the package components, in the pre-created directory structure mentioned above.

3. Create input `.xml` file with the required definitions.
4. Open the Command Prompt window. Enter the command-line as explained above.

Upgrade Snapshot Package

This feature provides an option to create an upgrade package from the command-line. Create the first version (version1) of the application using the GUI or command-line. An upgrade for this package can be created using the command-line. At a later point of time if you intend to create upgrade packages using the GUI, please ensure that the machine has the contents of both version 1 and version 2.

If a package has been upgraded through the command-line, exercise special care in order to use the Streaming Composer GUI for subsequent packages. Please ensure that the machine is the exact same state as version 1 and version 2.

Method 1-Upgrade package

Command line Syntax

Syntax:

```
Packager.exe /updatexml:<xml file> [/oldpackage:<old-package-path>]
<package-path>
```

where

`/udpdatexml:<xml file>`—Create an upgrade package using upgrade `.xml` file

`/oldpackage:<old package path>`—The location of previous version package zip file

`<package-path>`—The location where the upgrade package zip file is created

The command-line upgrades the existing package in the user specified location. All changes are made to this package. Hence, back up all production packages to a different location before the upgradation process

The upgrade `.xml` file contains package description properties that are used in the update package phase.

The Streaming Composer exits with the following exit codes:

- 0 - Success,
- 1 - Failure : Check log file for errors
- 2 - Package is invalid due to empty files content or to indicate that no changes have been found to create an upgrade package.

Note : Streaming Composer automatically increments the package version. This can be overridden by assigning a different version using the /d option. If the log file option was specified, it would contain details about the files that were replaced.

Example:

```
Packager.exe /updatexml:n:\Org\customer1\customer1.xml
n:\Org\customer1\package Org\upgradel\package
```

Streaming Composer creates the upgrade in the location n:\Org\upgradel\package using the information in the upgrade .xml file—n:\Org\customer1\customer1.xml and the original package contents in n:\Org\customer1\package.

Procedure

1. Create a directory tree structure as described in Source Files Path and the set root of this tree as the default “Source Root Folder”. Create this at the same hierarchical level as the files path defined for package creation.
2. Place the upgrade components (version 2), in the pre-created directory structure mentioned above.
3. Create update .xml file with the required definitions.
4. Open the Command Prompt window. Enter the command-line as explained above. Streaming Composer automatically detects the files that have changed from the previous version by comparing size and CRC data. After identifying all the changed files Streaming Composer updates the internal Blocks Database (DB1 file) and creates a new version of the package. For those files that have not been changed or do not exist in the source directory, Streaming Composer continues to use the existing file (version 1). For all the changed files, the content (blocks) of the version 1 file are replaced with the contents of the version 2 file.

Method 2-Upgrade package

This section illustrates the steps to be followed for upgrading a package from a package created using GUI.

1. Create first version (version1) of the application by the Streaming Composer GUI. Ensure that while creating version1 package, 'Source' folder is created inside the package path. To do this select the **Create Source Folder** check box in the **Package Information>Upgrade Package Properties** page. Make sure that this 'Source' folder contains the folder hierarchy of the package without any files, as discussed above.
2. Place the upgraded version of the application inside the 'Source' folder structure accordingly.
3. Run the Streaming Composer command-line

```
Packager.exe <Package Path> /update /log <log file name>
```

Streaming Composer identifies the list of changed files and creates a new version of the package.

Note : Back up the entire package folder to another location before running the upgrade.

Creating MSI Packages

MSI applications can be packaged using the Command-Line interface described in this section. This interface supports the creation of MSI major, minor, Patch and upgrade packages. The packages created using this interface, can be subsequently opened and upgraded using the Streaming Composer user-interface.

The packaging can be done in either of the following two modes :

Basic Mode

When this mode is used, Streaming Composer uses default configuration settings for all operations. Shortcuts and icons are automatically extracted by Streaming Composer.

1. Activate Command Prompt.
2. Change to the Streaming Composer directory

Example: `CD "C:\Program Files\Altiris\Streaming Composer"`

3. Enter the command-line:

```
Packager.exe <target package path> /createfrommsi/msisrc:<MSISourceFolder> /mainmsi:<MainMSIFileName>
```

Example:

```
Packager.exe C:\mypackage /createfrommsi/msisrc:c:\msi\myappmsisource /mainmsi:myapp.msi
```

This creates a package `mypackage` in `C:\` drive, from the input `myapp.msi` file.

Note : Remember to use quotes if there are spaces in the path.

Advanced Mode

This mode of packaging enables customization of the configuration of the application to suit the requirements of the target computers. Execution scripts, external executables, supported platforms and other such parameters can be included in the packaging process. Specify these additional custom parameters in a .XML file. The format of the .XML file format and its usage is similar to that described in “XML File” on page 67 .

1. Activate Command Prompt.
1. Change to the Streaming Composer directory

Example: `CD "C:\Program Files\Altiris\Streaming Composer"`

2. Enter the command-line:

```
Packager.exe <target package path> /createfrommsi/updatexml:<InputXMLFile>
```

Example:

```
Packager.exe C:\mypackage /createfrommsi/updatexml:C:\myfolder\newMSI.xml
```

Upgrade MSI Package

Basic Upgrade Mode

Upgrade packages can also be created using the command-line interface.

1. Activate Command Prompt.
2. Change to the Streaming Composer directory

Example: `CD "C:\Program Files\Altiris\Streaming Composer"`

3. Enter the command-line:

```
Packager.exe <DestinationFolder> /creatfrommsi /msisrc:<PathToMSI/MSPSrcFolder> /mainmsi:<MainMSIFileName> /oldpackage:<BasePackage-Path>
```

Example:

```
Packager.exe C:\MyUpgradeAppPackage /createfrommsi /msisrc:c:\msi\Appupgradesource /mainmsi:AppSP1.msp /oldpackage:C:\MyAppPackage
```

A MSI Patch package or a MSI Upgrade package is created depending on the mainmsi parameter i.e the name of the parent MSI file. Use `.msp` to create MSI patch and `.msi` to create MSI upgrade package. The packages are created in the `<DestinationFolder>`.

`/msisrc` is the full path to the MSI source folder. (If a self-extracting `setup.exe` file is used, then it should be pre-extracted to a folder before running this command line.) While using a self-extracting `setup.exe` file, pre-extract it to a folder before running this command line.

Advanced Upgrade Mode

This mode of package creation is recommended, when you need to customize the configuration of the application. Specify the additional custom parameters in a `.XML` file. The format of the `.XML` file and its usage is similar to that used while creating packages (using command-line interface).

1. Activate Command Prompt.
2. Change to the Streaming Composer directory

Example: `CD "C:\Program Files\Altiris\Streaming Composer"`

3. Enter the command-line:

```
Packager.exe <DestinationFolder> /creatfrommsi /updatexml:<InputXML-File> /oldpackage:<BasePackagePath>
```

Example:

```
Packager.exe C:\MyUpgradeAppPackage /creatfrommsi /upda-
texml:C:\Myfolder\UpgradeVSA.XML /oldpackage:C:\MyAppPackage
```

`/updatexml` is used in the advanced mode, to specify additional options in the `.xml` file (including custom setup parameters, execution scripts etc.). These are in addition to the `msisrc` and `mainmsi` parameters.

Once the streaming package has been created, upload the new package to a Streaming Server. Refer to the Admin Guide for details on this step.

Creating SVS Packages

Any `.VSA` file can be converted for subsequent streaming on streaming enabled machines. To create SVS packages, adopt the command-line mode of operation described below. The packaging can be done in either of the following two modes :

Basic Mode

In this mode, Streaming Composer uses default configuration settings for all operations. Shortcuts and icons are automatically extracted by Streaming Composer.

1. Activate Command Prompt.
2. Change to the Streaming Composer directory

Example: `CD "C:\Program Files\Altiris\Streaming Composer"`

3. Enter the command-line:

```
Packager.exe <target package path> /createfromvsa:<path to vsa file>
```

Example:

```
Packager.exe C:\mypackage /createfromvsa:C:\test\myapp.vsa
```

This creates a package `mypackage` in `C:\` drive, from the input `myapp.vsa` file.

Remember to use quotes if there are spaces in the path.

Advanced Mode

Use this mode to customize the configuration of the application to suit the requirement of the target computers. Specify execution scripts, external executable, supported platforms and other such parameters as a part of the packaging process. Specify these additional custom parameters in a `.XML` file. The format and usage of `.XML` file is similar to that used while creating packages (using command-line interface), as described in “*XML File*” on page 67.

1. Activate Command Prompt.
2. Change to the Streaming Composer directory

Example: `CD "C:\Program Files\Altiris\Streaming Composer"`

3. Enter the command-line:

```
Packager.exe <target package path> /createfromvsa:<path to vsa file>  
/updatexml:<path to XML file>
```

Example:

```
Packager.exe C:\mypackage /createfromvsa:C:\test\myapp.vsa /upda-  
texml:C:\myfolder\newVSA.XML
```

Once the streaming package has been created, upload the new package to an Streaming Server. Refer to the Admin Guide for details on this step.

Summary

This chapter has described the various Packaging functionalities that can be handled using the Command-Line Interface.

Appendix A : Execution Script-An Example

This section is a walkthrough of the process of configuring an execution script with Streaming Composer.

Script to check if current version of IE is 6.0 or higher

A sample script is given below, checks if the Internet Explorer currently available on the system is version 6.0 or higher.

```
@echo off

REM *****

REM This batch file checks if Shdocvw.dll has a major version greater
than or equal to 6.

REM This version determines the version of IE installed on the machine.
refer to : http://support.microsoft.com/default.aspx?scid=kb;en-us;164539

REM parameters:

REM resultfilename - The output will be written to this file.

REM filename - whose version has to be checked.

REM *****

REM Check the version of the given file.

apputil -filever %2>findapp.bat

REM The application found in add/remove program list is printed to find-
app.bat

call findapp.bat

REM If this is empty, that means this file was not found OR the file ver-
sion could not be determined.

if "%AS_FILE_MAJOR_VERSION%"==" " goto FAIL

if %AS_FILE_MAJOR_VERSION% GEQ 6 goto PASS
```

```

:FAIL

REM Used pressed "no",write 0 indicate failure

REM Prompt the user to uninstall the application.

apputil -ask "Internet Explorer 6.0 or higher is required to run MSMoney.
It is recommended to upgrade your browser and try again. Would you still
like to proceed?"

REM user pressed 'yes'

REM if errorlevel 1 goto PASS

echo 0 >%1

goto end

:PASS

REM Write the result to the result file

echo 1 >%1

REM End the batch script

:end

```

Add this script to Streaming Composer so that the client would invoke it before the virtual installation of the application.

1. Navigate to **Package Information>Execution Scripts**. Click **Add** to open Add/Modify Execution Script page.
2. Fill the parameters as described below:
 - **Script Type**—Determines when this script would be executed.
 - **Script**—The name of the batch file that calls the script.
 - **Parameters**—Command-line parameters for the batch file/executable. `$script-result$` is a mandatory parameter. At run time, the Streaming System substitutes this with the actual result file. The script should write the return code into this file. A return code 1 indicates success and 0 indicated failure. The script can include a descriptive error message in the second line. The Streaming Agent shows this error message, if there is a failure.
3. Since another script `apputil` is called by this batch file, it is essential to describe the script name in a separate Add/Modify Execution Script page as a Data.
4. Fill the parameters as:
 - **Number**—The next consecutive number.
 - **Script Type**—**Data** to indicate that this script is executed through a batch file, and not directly by the Streaming Composer.
 - **Script**—Script name complete with the path name.

Execution Script Variables

Several variables have been provided for an advanced scripting functionality.

:

Remember to pass these variables to the scripts from the command-line. This is important because the script itself is not parsed for these variables by the client.

Example :

```
Myscript.bat $scriptresult$ $APPS_PATH$ $ASPackage_GUID$....
```

Variables

These variables are used by Streaming Composer, but are made available specifically for use in execution scripts.

\$APPS_PATH—Variables already built into Streaming Composer.

“ASPackage_GUID” — replaced by a string containing the package guide (no dashes, 32 characters).

"ASPackage_ID"—replaced by a string in decimal format.

"ASPkgInstalledVersion"— Used in an upgrade package, provides the version of the package already installed.

"ASPkgInstallingVersion" — The version of the current package.

"ASPkgInstalledVendorVersion"— The vendor version of the package already installed.

Script to uninstall any MSI installation silently

A sample script is given below, silently uninstalls any MSI application :

```
@Echo Off

REM This batch file silently uninstalls the given MSI application

set RESULT_FILE=%1

MsiExec.Exe /x <product-id> /qn

if "%ERRORLEVEL%"=="0" goto SUCCESS

if "%ERRORLEVEL%"=="1605" goto REMOVED

goto FAIL

:FAIL

Echo 0 > %RESULT_FILE%

Echo Uninstalation Failed >> %RESULT_FILE%

goto END
```

:

:REMOVED

Echo 1 > %RESULT_FILE%

Echo Application Not Installed >> %RESULT_FILE%

goto END

:SUCCESS

Echo 1 > %RESULT_FILE%

Echo Uninstalation Success >> %RESULT_FILE%

:END

Appendix B: AppstreamCfg File

```
;Debug.GenTraceLevel=DEBUG_INFO
Debug.GenTraceLevel=INFO
Debug.LogFile.TxtPath=$(packagerpath)\Packager.log
Debug.LogFile.DebugView=0
Debug.LogFile.BuildXsl=0
Debug.LogFile.UseXsl=0
Debug.LogFile.BytesSize=500000
Debug.LogFile.DecoratingPolicy=0

; License Information
License.File.Path=2005_12_01_AppStream_Internal_Pkg.lic

; For Packaging Tool
ActivateFileMonitorDriver=1
TimeoutFileMonitorDriver=3
CompressBlocks=YES
```

:

Appendix C : Packager.INI File

```
[packager]
packager.packages.path=D:\FolderShare_1_XP 5.0x
packager.navigation.autonextstep=true
packager.repository.path=
packager.cm.url=
packager.engine.format=50
packager.engine.excludedeleted.file=false
packager.engine.excludedeleted.registry=false
packager.engine.excludedeleted.ini=false
packager.engine.excludedeleted.text=false
packager.engine.ShowParseDlg.ini=false
packager.display.ghost=false
packager.config.active=Default
packager.shortcut.force.import=false
packager.shortcut.force.user=false
packager.build.force=false
packager.image.draw.disable=false
packager.engine.sidebyside.default=false
packager.build.zip.compress=false
packager.build.zip.skip=false
packager.build.zip.UptoLimit=2
packager.msi.upgrade.reusefileid.skip=false
packager.msi.upgrade.reuseonlydest=false
packager.msi.upgrade.reuseguess=false
```

:
packager.msi.default.parameters=/q \$INSTALLDIR="\$APPS_PATH\$" ALLUSERS=2
packager.msi.default.parameters.uninstall=/q
packager.msp.default.parameters=/q REINSTALL=ALL REINSTALLMODE=omus
packager.create.truegif=false
packager.checksize.package=100
packager.checksize.application=100
packager.snapshot.import.guessroot=false
packager.name.minlength=5
packager.isshield.iscab.path=C:\Program Files\InstallShield\InstallShield Professional
6.2\Program\ISCAB.exe
packager.isshield.default.parameters=-SMS -s -f1"\$APPS_PATH\$\as_isfiles\INSTALL-
RESPONSEFILE\$" -f2"\$LOGFILE\$"
packager.isshield.default.silentuninstall.parameters=-SMS -s -f1"\$UNINSTALLRESPON-
SEFILE\$" -f2"\$LOGFILE\$" -verbose
packager.isshield.default.uninstall.parameters=-SMS -f2"\$LOGFILE\$" -verbose
packager.isshield.upgrade.reusefileid.skip=false
packager.isshield.upgrade.reuseonlydest=false
packager.maxscript.size=10
packager.maxappinfo.size=12
packager.cmdline.createdir=true
packager.cmdline.defaultpath=Source
packager.createdb.tempfolder=false
packager.window.location=-4 -4 1028 744
packager.window.pane=165 848 577 71

[packager.name.disallow]
packager.name.disallow.1=admin
packager.name.disallow.2=administrator

:

packager.name.disallow.3=client

packager.name.disallow.4=server

packager.name.disallow.5=user

[platforms]

platforms.1=Windows NT 4.0=Windows NT 4.0

platforms.2=Windows NT 5.0=Windows 2000

platforms.3=Windows NT 5.1=Windows XP

platforms.4=Windows NT 5.2=Windows 2003 Server

[msi-setup-launcher]

msi-setup-launcher.1=Description=*Launcher*|Content=*InstallShieldMSI*||/w /s /v"/q
\$INSTALLDIR\$="\$APPS_PATH\$" ALLUSERS=2 \$MSIPARAMS\$"

msi-setup-launcher.2=Description=*Bootstrapper*|Copyright=*Microsoft*||/wait /q
\$INSTALLDIR\$="\$APPS_PATH\$" ALLUSERS=2 \$MSIPARAMS\$"

msi-setup-launcher.3=Content=*WiseForWindowsInstaller*||/q \$INSTALL-
DIR\$="\$APPS_PATH\$" ALLUSERS=2 \$MSIPARAMS\$"

[msp-setup-launcher]

msp-setup-launcher.1=Version=*|Description=*Launcher*|Copyright=*InstallShield*||/w
/s /v"/q \$INSTALLDIR\$="\$APPS_PATH\$" \$MSIPARAMS\$"

[msi-setup-sfx]

msi-setup-sfx.1=Content=*InstallShield.setup*||/a /s /v"/qn TARGETDIR="\$TARGET-
DIR\$""

msi-setup-sfx.2=Content=*WiseForWindowsInstaller*||/qn TARGETDIR="\$TARGET-
DIR\$" /a

[snapshot-install-msi-setup-launcher]

:

```
snapshot-install-msi-setup-launcher.1=Description=*Launcher*|Content=*InstallShieldMSI*||/v"ADDLOCAL=ALL DISABLEADVTSHORTCUTS=1"
```

```
snapshot-install-msi-setup-launcher.2=Description=*Bootstrapper*|Copyright=*Microsoft*||ADDLOCAL=ALL DISABLEADVTSHORTCUTS=1
```

```
snapshot-install-msi-setup-launcher.3=Content=*WiseForWindowsInstaller*||ADDLOCAL=ALL DISABLEADVTSHORTCUTS=1
```

[ishield-setup-launcher]

```
ishield-setup-launcher.1=Description=*Launcher*|Content=*\InstallShield\Engine\6\Intel 32\*||-SMS -s -f1"$APPS_PATH$\as_isfiles\$_RESPONSEFILE$" -f2"$LOGFILES"
```

Appendix D : Appstream.INI File

The `Appstream.INI` file is used as a conduit to convey information about the package to the Streaming Agent. It is created automatically during the last stage of the packaging process. However, certain functionalities can be extended, by manually adding additional information to this. Its structure and possible values are given below.

1. **Variables**—Contains custom, package specific variables. The user can use it to specify custom variables to be used in execution scripts.
2. **PackageDescription**—Contains description about the package including applications etc. This is filled in automatically by Streaming Composer during the last step. (“create package”).

The only option not available currently from the GUI is: `IgnoreWFPCheck`

The default value is false. However, it is recommended that this value is not changed.

If a package has lot of shortcuts and you wish to display only one application on the Launch page, then specify,

```
InstallationType=mainapp:{guid}
```

3. **DynamicFiles**—Specifies the dynamic files and data that must be replaced including the offset. This is filled in automatically by the Streaming Composer during the last step.
4. **ScriptsExecutable**—Contains a list of all the scripts and executables.
5. **PreInstallScripts**—Data about the pre-install scripts
6. **MSI**—MSI related information.
7. **Fonts**—Data about fonts.
8. **MSIPatches**—Data about patches.

Patches include an “attribute” field which can be used to specify additional values. Currently, only “reboot” value is supported and this instructs the Streaming Agent to reboot the machine after applying that patch.

For example : `ApplicationAttributes.1=Reboot`

9. **ProcessHooking**—Specifies instructions to the Streaming Agent to handle hooking. These include list of processes that should be hooked, should not be hooked, should not be suspended, etc.

All the settings are for the current package only. These only need to be specified if they are not listed in the Streaming Agent configuration file.

- **ProcessName**—Specify additional processes that must be hooked for this package.
- **LongLived**—Specify processes that run continuously in the background. Streaming Agent hooks these but closes the session even if they are still running.
Adobe acrobat is a good example for this. It has two processes, `acrotray.exe` and `acrodist.exe` that are long lived.
- **Unsuspendable**—List of processes that should not be suspended. For example, use it for Babylon. If it is suspended due to any reason (for example, when there is a network outage) it locks up the entire machine.
- **Ignored**—List processes that should not be hooked.

Specify only the process name, the entire path need not be specified.

Example:

The format for all the other options is similar to this.

```
ProcessName.1=someprocess1.exe
```

```
ProcessName.2=someprocess2.exe
```

Appendix E : How to Deploy Service Packs & Hotfixes

This chapter describes how to create a package for Operating System Service Packs and Hotfixes. These packages would be deployed as pre-populated packages, enabling administrators to “push” the service packs and hotfixes onto Streaming Agent machines. The following sections describes the procedure in detail.

- *The Create Phase*
- *The Package Content Phase*
- *The Package Information Phase*
- *Building the Package*
- *Package Deployment*

The Create Phase

In most cases, create a package on a clean system because you can then deploy the application to most desktop systems. However, if each computer that you plan to install the package on has the same image, then create the package on a system with the common image.

The Create phase consists of the following steps:

- Create a folder in the System drive for patch package. For example `E:/WinXP_SP2`.
- Include this folder for monitoring
- Take a baseline snapshot of the system. This system mirrors the system where the application may run.
- Add a `.bat` file in the folder created above. This acts as a trigger to launch the application on the user desktop.
- Take a post-setup snapshot of the system.
- Process the difference between the system snapshots.

To Startup

1. Create a folder in the System drive for patch package. For example `E:/WinXP_SP2`.
2. Navigate to the Streaming Composer on the Windows Start menu: **Start>Programs>Altiris>Streaming Composer**. Execute the application.

To Customize Environment

It is very essential to monitor the file system in which you plan to install the application. By default, it is setup for drive C:.

To setup the drive that suits your requirements :

1. Navigate to **Tools>Packaging Options>Monitoring>Files**.
2. The Monitored Files window displays the list of drives/folders that are monitored. You can **Add** new drives/folders, **Modify/Remove** the path(s) displayed here.
3. Click **Add**. Folder/Drive Selection window is displayed. Browse or enter the folder created in step 2 of Startup. (For example winXP_SP2 as given above).

To Create a Package

1. Click the **Create New** tab in the New/Open Package dialog box or select the **New** option from the **File** menu.
2. On the Create New page displayed, choose the **Snapshot Package** option.
3. The completed package resides in the **Package Path** folder. Enter the folder path or browse to the designated package folder. This path should not be the same as that of the path of the current operating system. It must be complete and cannot be a network path. For example, F:\WinXP_Service Pack 2\.
4. The **Package Name** is the name you assign to the package. Enter a name. It is strongly recommended that the name describes the Application and the OS it is intended to run on.
5. Set configuration profile to **Default**.
6. Click **OK** and the Create Application screen is displayed, to take you through the steps of creating the snapshot package.

Note : When you log on with a view to creating a new package, ensure that the user name and computer name are significantly different. If these are not different, while creating a new package you see an error message - *To avoid packaging problems, user name and computer name should be significantly different*. Do not log on as Administrator, while creating a package.

To Take a Baseline Snapshot

1. The **Baseline Snapshot** step records the state of the system before you install the application.
2. Click **Baseline Snapshot** under the **Create** menu in the Navigation pane. Take or Import Baseline Snapshot screen is displayed.
3. Click **Run** to take the snapshot. The Streaming Composer scans the file system for relevant files and registry entries. This process may take a few minutes.

Install Application

After completing the baseline snapshot, Streaming Composer automatically displays the Install Application page.

Copy the application executables (like `setup.exe`) to the folder created in step 1 of Startup. You must also copy the `.bat` file (created to trigger the launch of the application on the user desktop) and the readme files, if any, into this folder.

To Take a Post Installation Snapshot

1. The **Post-Setup Snapshot** page is displayed. The Navigation Pane and the Status Pane display this step.
2. Click **Run**. The Streaming Composer scans your file system for relevant installed objects and takes the post-setup snapshot of the files, registry entries after the application is installed. The length of the scan depends on the number of files and registry entries on the system or path.

To Process the Difference Between Snapshots

In most cases, when you process the difference between the snapshots, the package meets all requirements and you can continue to examine the package content.

1. The **Process Differences** page is displayed.
2. Select or enter the correct **Root Path** for your application. The Streaming Composer streams files that are on the Root Path directory or its subdirectories to the Streaming Agent cache. Files that are not installed on the client by the Streaming Composer, remain outside the cache.
3. Click **Run**.

The Package Content Phase

At this point the Streaming Composer assumes that the File System has been configured as required by the application. Consequently, it moves forward to the "Generate Package Streamlets" window. However, you need to review the Program and Data Files.

Review Program and Data Files

To Review Files, Registry Entries, and INI Entries

1. Navigate to **Package Content>Files**
2. Select an entry from the drop-down list.
3. Use the tree view to navigate through the elements. As you select each element, the **Files** associated with it are exposed in the lower table. The **Path** field shows your selected node.

- :
4. By default, the AS Attributes are set as "S". Set AS Attributes as "I" for the folder set for monitoring. Do not modify the attributes for the .bat file and readme.txt (if any). Refer to “*Review File, Registry, and INI Files*” on page 29 for more details

Package Environment

Package Content>Package Environment rebuilds the package environment, and saves your changes to the Package Content. If you change any files, then you must rebuild the package. Click **Update** to rebuild the package.

The Package Information Phase

Use the Package Information Phase to define execution scripts and applications, split the application into streamlets, and collect data for the startup block.

Define Execution Scripts

Package Information>Execution Scripts is used to define scripts that the client executes for some milestones. Include a post-installation script to implement silent installation of the Service Pack/Hotfixes.

Click **Add** to open Defining Execution Scripts screen.

To Add a Script

1. Navigate to **Package Information>Execution Scripts**.
2. Click **Add**. Select a script type (described below) from the drop down list and complete the optional parameters.
 - **PreInstallation Script**—Add a batch file to determine, if the OS Service Pack or the Hotfix for which a package is being created (for example `Windows XP SP2`), has been installed conventionally on the current system.
 - **Post Installation Script**—Add a batch file to install OS Service Pack or the Hotfix, conventionally on the current system silently.

Update the Application List

Package Information>Application displays a list of program executables that the Streaming Composer recognized in the application. You can specify additional executables, remove executables, or modify the attributes of currently defined executables. Add the `Readme.txt`, if required.

To Add New Applications

You can specify additional executables which can also be available as shortcuts in the Windows Start menu.

1. Navigate to **Package Information>Application**. Update Application List is displayed.
2. Click **Add**. Enter details in the Add/Modify Application window using the following definitions to define an executable :
 - **Unique ID**—A unique identifier for the application, automatically provided by the Streaming Composer.
 - **Version**—An internal version number generated by the Streaming Composer, to indicate the version of the application.
 - **Name**—The name of the application or executable within the package. The application vendor usually supplies this name.
 - **Path**—Select the `.bat` (described earlier) from Path combo box.

Generate Streamlets

Package Information>Streamlets section creates streamlets for the packaged application. Streamlets are the 4KB blocks that are transferred from the Streaming Server to the client. The Streaming Composer completes the values in the table once you create the streamlets. This step may take a few minutes, depending on the size of your application. To generate the streamlets, click **Run**.

Collect the Package Startup Block

Package Information>Startup Block creates a startup block for the packaged application. The startup block is a collection of streamlets required to start the application.

Click **Skip** to bypass collection of startup blocks now, and allow the Streaming Console to perform this activity once the package is deployed on the server.

Building the Package

Use this phase to build the package. The Streaming Composer collects the generated streamlets and the startup block and creates a zip file.

To Build the Package

1. Navigate to **Build>Package**.
2. Click **Create**. The Streaming Composer builds the package. If the Streaming Composer finds orphan paths, it displays the Build Process Manager dialog box. Refer to the next section to resolve such orphan paths.
3. Click **OK** in the dialog box that pops up when the package is complete.
4. Click **Launch Streaming Console** to upload this package to the Streaming Console. To use this feature, you must specify the Control and Management URL in the Packaging Options.

Refer “*Building the Package*” on page 38 for more information.

Package Deployment

To deploy the package which has been created, upload the package using Streaming Console. Refer *6.3.2 Add Packages to the Streamlet Engine in Administrator’s Guide* for procedural details.

Provision the package to the users, such that the package pre-populates the client desktop. The procedure is given in detail in *6.4 Provisioning Packages and Applications in Administrator’s Guide*.

Summary

The various steps involved in creating a package operating system service packs and hot-fixes has been described in detail in this chapter.

Glossary

Agent—A component of the Streaming system that the Streaming Console uses to control, monitor, and manage Streaming Servers and licenses.

Streaming Agent—The part of the client that controls the flow of events in the streaming process including negotiation between the server and Client, execution of the streaming process, and application termination. Its graphical user interface, viewable on client systems, displays information about sessions, servers, and streamed applications.

Client—The client allows users to automatically stream applications. The first time a user accesses a streaming application, the Streaming Server automatically installs the Client. The Client contains the Streaming Agent.

Streaming Agent Cache—The Streaming Agent cache stores streamlets accessed from the Streaming Server for quick access. The critical components of the client are the persistent cache file system on the hard drive and a private registry entry in the user's registry.

Streaming Agent Cache File System—In a streamed application, access to the computer's file system is rerouted to the persistent Streaming Agent cache. This cache acts as a sparsely populated file system because only those streamlets that the application requires are kept there. Unreferenced streamlets remain on the server, conserving network bandwidth and disk space. The cache remains persistent between sessions.

Streaming Console—A web-based application for administering the Streaming System. You can configure and monitor all of the Streaming Servers, configuration groups, and users. When you install the Streaming Console, you also install a command-line tool with basic functionality. You can use it to install and configure local or remote servers that do not support monitors, GUIs, or console access.

Installable Files—Files that cannot be stored in the Streaming Agent cache (such as DLL files and configuration files that must be installed in a system folder).

Launch Server—Generates dynamic HTML pages for the user to see applications provisioned by the system administrator.

MSI Package—Applications created with Windows Installer/MSI should only be packaged as a MSI package. The logic available with the application installer is used to set up the entire system at the time of streaming the application. The entire package will be provisioned.

MySQL—The database used by the Streamlet Engine.

Package—A set of files that represents the complete application, which the Streaming Composer has prepared for streaming.

Predictive Learning—The Streaming Server uses predictive learning to determine what parts of an application a user is about to use next based on prior user behavior. It then streams the predictive parts of the application to the users desktop.

Server Group—This setup allows you to configure all the servers in the set or farm at once. A server group can contain multiple Streaming Servers and Launch Servers, but only one Streamlet Engine.

Snapshot—When you create a Snapshot package, you take a Snapshot of the system, install the application, take a second Snapshot, and compare the two Snapshots. The Streaming Composer processes the difference between the two Snapshots to collect the contents of the Package.

Startup block—The minimum set of streamlets that the Client needs to begin executing the application.

Streaming Server—The component that efficiently streams the application to Clients.

Streamlet— Streaming technology software segments applications into smaller components called streamlets. The Streaming Server software stores these streamlets in a repository. When a user opens an application from a package, the server streams these streamlets to the user's desktop.

Streamlet Engine—The Streamlet Engine provides centralized data storage for streamlets, configuration, and other user data. It also functions as a central data storage in a load balance environment. It can work with data from a Lightweight Directory Access Protocol (LDAP) repository, MS SQL database, an NT4 Domain or from the Streaming System's internal MySQL database.

SVS Package—Some applications can be virtually installed on the client desktop using the virtualization facility offered by Altiris. The Streaming System supports package creation of .VSA files created using Altiris Software Virtualization Solution. Such packages can be subsequently streamed and monitored for license usage.

Tracking Package—Tracking packages enable tracking the usage of applications that have been conventionally installed on client desktop. By default, they are offline access enabled.

Virtual Installation—To stream an application, a user accesses a web page called the Launch page. This page displays a list of links for each package or application that the administrator has provisioned to this user. When the user clicks a streamed application or a pre-populated icon, the client performs a virtual installation on the user system. The virtual installation retrieves the application from the server, and includes all the information that the application needs to be streamed.