

Publisher Guide

Wyse WSM™ Release 3.0

Issue: 011310

PN: 883876-02 Rev. A

WYSE
| | | |

Copyright Notices

© 2010, Wyse Technology Inc. All rights reserved.

This manual and the software and firmware described in it are copyrighted. You may not reproduce, transmit, transcribe, store in a retrieval system, or translate into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, any part of this publication without express written permission.

End User License Agreement (“License”)

A copy of the Wyse Technology End User License Agreement is included in the software and provided for your reference only. The License at <http://www.wyse.com/license> as of the purchase date is the controlling licensing agreement. By copying, using, or installing the software or the product, you agree to be bound by those terms.

Trademarks

The Wyse logo and Wyse are trademarks of Wyse Technology Inc. Other product names mentioned herein are for identification purposes only and may be trademarks and/or registered trademarks of their respective companies. Specifications subject to change without notice.

Restricted Rights Legend

You acknowledge that the Software is of U.S. origin. You agree to comply with all applicable international and national laws that apply to the Software, including the U.S. Export Administration Regulations, as well as end-user, end-use and country destination restrictions issued by U.S. and other governments. For additional information on exporting the Software, see <http://www.microsoft.com/exporting>.

Ordering Information

For availability, pricing, and ordering information in the United States and Canada, call 1-800-GET-WYSE (1-800-438-9973) or visit us at <http://www.wyse.com>. In all other countries, contact your sales representative.



Contents

- 1 Introduction 1**
 - About this Guide 1
 - Organization of this Guide 1
 - Wyse Technical Support 2
 - Related Online Resources Available at Wyse 2
 - Wyse Online Community 2
- 2 Publishing Overview 3**
 - Publishing Overview 3
 - Publishing Steps 5
 - WSM Publisher File Types 8
- 3 Getting Started 9**
 - Hardware and Software Requirements 9
 - About Publishing Performance 9
 - Preparing a Dedicated Machine 9
 - Setting-up Disk Space for Publishing 10
 - Installing WSM Publisher 10
- 4 Creating Build Files 11**
 - Creating a Build File Using Project Wizard 11
 - About a Project 11
 - Creating a Build File Manually 15
 - Saving the Build File 20
- 5 Modifying Build Files 21**
 - About the Publisher Build File Window 22
 - Undoing and Redoing Changes 22
 - Undoing the Last Change 23
 - Undoing Multiple Changes 23
 - Redoing the Last Undone Change 23
 - Redoing Multiple Undone Changes 23
 - About Cleaning Up Build Files 24
 - Using Macros for Automated Clean-up 24
 - Cleaning-up the Build File Manually 26
 - Build Tree Structure 26
 - Files Sub-tree 26
 - Registry Entries Sub-tree 27
 - Clean-up Operations Overview 27
 - Clean-up Guidelines 27
 - Where to find Additional Help and Information 28
 - Inserting, Removing, Including and Excluding Entries 28
 - Inserting a Registry Root 29
 - Inserting a File or Folder 30
 - Inserting a Registry Key or Value 30
 - Removing, Including, or Excluding an Item 31

Generating, Adding, and Pruning Reports	31
Generating a Report of Included and Inserted Entries	31
Generating a Report of Removed and Excluded entries	31
Resetting Tracking for a Report	32
Modifying Registry Values and Types	32
Changing Registry Values	32
Changing Registry Value Types	33
Recording Macros	33
Recording Changes in a Macro Session Log	34
Setting Access Tokens	35
Changing File and Folder Dispositions	36
Editing a File or Folder Disposition	37
Viewing Change Types	37
6 Generating Appsets	39
Creating an Appset	39
Encrypting and Decrypting Appsets	46
Encrypting an Appset	46
Decrypting an Appset	47
Viewing Appsets	47
Viewing the Contents of an Appset	47
Extracting a File from an Appset	48
What's Next	48
7 Testing and Optimizing	49
Loading Appsets	49
Testing Appsets	50
Delivery Testing	50
Application Testing	50
Correcting Common Problems	51
Desktop/Start Menu Icons Missing	51
Microsoft Icons do not Appear	51
File Associations not Set Properly	51
Command Path Broken	52
No Print Output or Printing Problems	52
System Fonts Appear Corrupted	52
Microsoft Installer Pop-ups Display	52
Missing Files	53
Application Licensing Problems	53
Miscellaneous Errors	53
Creating and Using a Prefetch File	53
Cleaning Up the Client Cache	54
Creating a Prefetch File	54
Republishing an Appset with a Prefetch File	55
Modifying and Updating Appsets	56
Modifying the List of Supported Operating Systems	56
Adding, Removing, or Modifying AppEvent DLLs	56
Republishing an Application	57

8	Publisher Macros	59
	Macro Example	59
	Creating and Editing Publisher Macros	60
	Editing a Macro Using the Built-in VBScript Editor	60
	Macro References	61
	CurrentDoc Object	61
	CurrentDoc Methods	61
	AddDirTree Method	62
	AddFile Method	62
	AddReg Method	63
	AddRegTree Method	64
	DeleteFile Method	64
	DeleteReg Method	65
	ExcludeFile Method	66
	ExcludeReg Method	67
	GetFiles Method	68
	GetRegType Method	68
	GetRegValue Method	69
	GetRegValueNames Method	70
	GetSubDirs Method	70
	GetSubKeys Method	71
	SetAccessToken Method	72
	SetAimDisposition Method	72
	SetRegType Method	73
9	Case Studies	75
	Publishing Macromedia Dreamweaver MX	75
	General Requirements for Publishing	75
	Preparing for Publishing	75
	Taking Snapshots	76
	Installing Dreamweaver MX	76
	Cleaning Files in the .aeb File	76
	Cleaning Registry Settings in the .aeb File	78
	Conclusion	79
	Publishing Microsoft Word 2003	80
	General Requirements for Publishing	80
	Preparation	80
	Installing Word	80
	Cleaning Files in the .aeb File	81
	Cleaning Registry Settings in the .aeb File	83
	Conclusion	85
A	Handling AppEvents	87
	AppEvent Types	88
	AppEvent Handlers	88
	Handler Configuration	89
	Configuration Macro	91
	Environment Variables	91
	Handling AppEvent Example	91
	Publishing Instructions	92
	Including Handler Executables in the Appset	92
	Setting-up the Registry Configuration	92
	Adding the CAED to the Appset	92

Figures	93
----------------	----

Tables	95
---------------	----

This page intentionally blank.



1

Introduction

WSM is an operating system and application distribution and streaming technology that provides a cost-effective method to rapidly deploy and manage operating systems and applications to a large pool of users. This guide describes how to use WSM Publisher to publish applications and to make them available for distribution. Throughout this document, WSM Publisher is also referred to as the Publisher.

About this Guide

This guide is intended for administrators of the WSM system. Following are guidelines for recommended skills and knowledge in Information Technology (IT), PC platforms, and Operating Systems (OS) technologies for personnel working with WSM Publisher:

- Working knowledge of Windows platforms: Windows XP and later (including the differences between the platforms and the installation of service packs).
- Working knowledge of hard disk partitions and renaming.
- Working knowledge of basic, distributed, and networking file-system aspects of the Windows OS, including the organization of Program Files, Roaming profiles, User start menus, and system folders.
- An understanding of DLLs, program executables, shared libraries, and application parameters.
- Working knowledge and understanding of the Windows registry, including machine profiles, user profiles, and software registry entries.
- Working knowledge of setting-up and configuring Windows applications, such as Microsoft Office.
- Understanding of application packaging (for example, InstallShield AdminStudio) is helpful.
- Experience with an automated/centralized networking tool (for example, Tivoli, Openview, or Unicenter).
- Understanding of the Java Apache Tomcat Web server.
- Working knowledge of a network monitoring tool.

Organization of this Guide

This guide is organized as follows:

- Chapter 2, "Publishing Overview," provides an overview of the WSM Publisher process.
- Chapter 3, "Getting Started," contains hardware and software requirements and the procedures you must complete to install the WSM Publisher.
- Chapter 4, "Creating Build Files," provides the detailed procedures you need to create a build file.
- Chapter 5, "Modifying Build Files," contains the procedures you need to clean up and fine tune a build file.

- Chapter 6, "Generating Appsets," provides the detailed procedures you need to create an appset.
- Chapter 7, "Testing and Optimizing," contains the procedures you need to test and optimize an appset. It also contains information on correcting common problems that may occur with appsets.
- Chapter 8, "Publisher Macros," describes the macros provided with WSM Publisher.
- Chapter 9, "Case Studies," provides two case studies to demonstrate the publishing and cleanup procedures and processes applied to specific applications.
- Appendix A, "Handling AppEvents," provides important information on handling AppEvents.

Wyse Technical Support

To access Wyse technical resources, visit <http://www.wyse.com/support>. If you still have questions, you can submit your questions using the [Wyse Self-Service Center](#) (on the Wyse.com home page, go to **Support | Knowledge Base | Home** tab) or call Customer Support at 1-800-800-WYSE (toll free in U.S. and Canada). Hours of operation are from 6:00 A.M. to 5:00 P.M. Pacific Time, Monday through Friday.

To access international support, visit <http://www.wyse.com/global>.

Related Online Resources Available at Wyse

Getting Started Guide: Wyse WSM Appliance™ is intended for administrators of the WSM system. It provides a setup and configuration overview of the entire WSM system to help you get your WSM environment up and running quickly and easily.

Installation Guide: Wyse WSM™ is intended for administrators of the WSM system. It describes the WSM installation process for Windows servers and clients. This guide provides the step-by-step instructions you need to install and configure a WSM environment. It also includes the requirements you must address before you begin the installation procedures.

Administrators Guide: Wyse WSM™ is intended for administrators of the WSM system. It provides information, and detailed system command and parameter configurations, to help administrators design and manage a WSM environment. It also explains how to use WSM, manage the availability of software applications for distribution to subscribers, manage application subscription licenses, install and configure published applications, provide subscriber profile and billing information for efficient application usage tracking, and control subscriber access to the WSM system.

Users Guide: Wyse WSM™ is intended for users of the WSM Client system. It provides detailed instructions on using the WSM Client to manage the applications available to users from a network server.

Wyse Thin Computing Software is available on the Wyse Web site at: <http://www.wyse.com/products/software>.

Wyse Online Community

Wyse maintains an online community where users of our products can seek and exchange information on user forums. Visit the Wyse Online Community Forums at:

<http://community.wyse.com/forums/>



2

Publishing Overview

This chapter provides an overview of the WSM Publisher process.

Publishing Overview

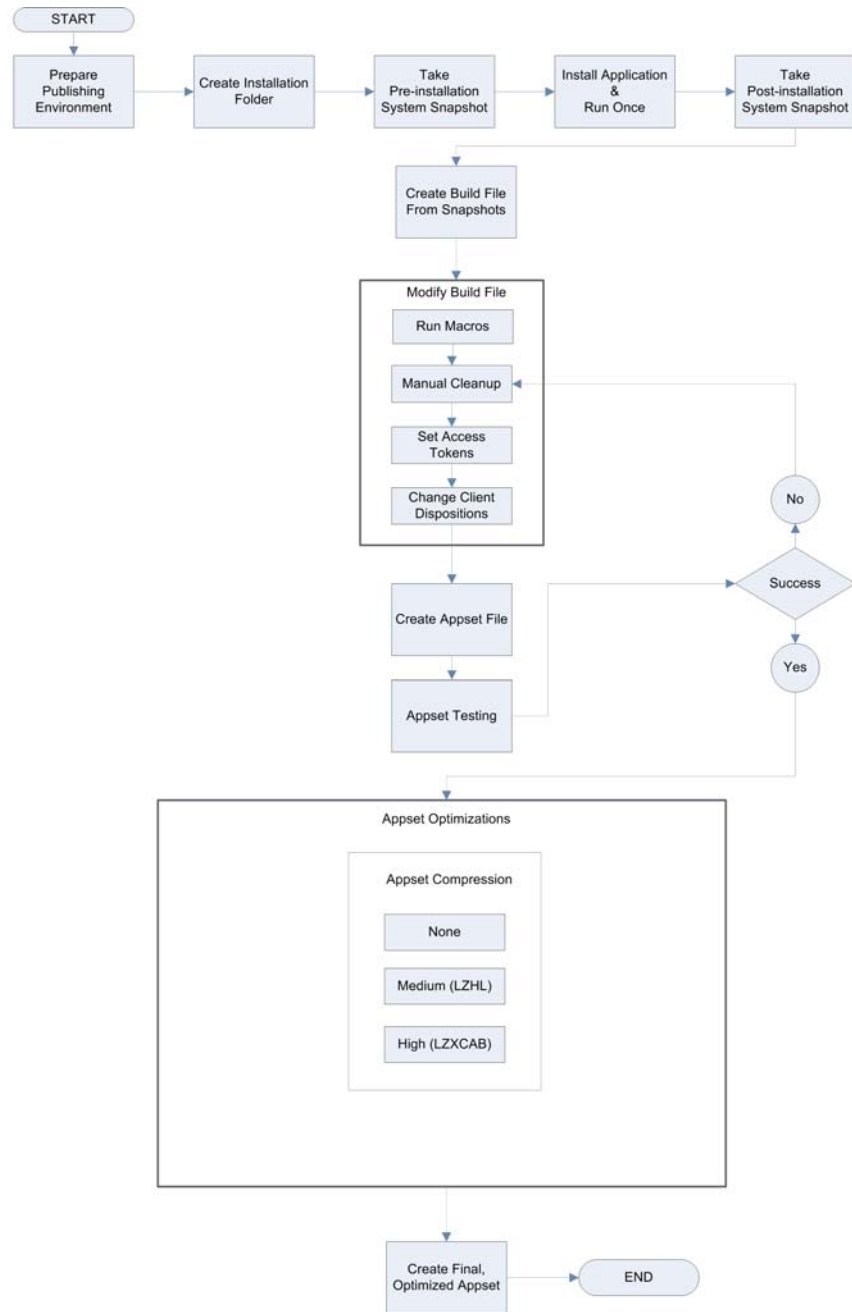
Publishing means to re-package a software application into a form that can be streamed to client machines. To publish an application, it is first installed on a platform that is dedicated to publishing and has WSM Publisher installed. WSM Publisher (also referred to in this guide as Publisher) determines how the installed application works in the Windows environment and packages the required components and configuration information into a compressed, encrypted application set, called an *appset*. The application being published is referred to as the *target application*. The appset is loaded on WSM servers and the target application becomes available for clients through streaming from the WSM servers.

The Publisher determines the required components and configuration for the appset by comparing snapshots of the system state prior to and after the target application is installed on the publishing machine. The differences identified in the files and settings are stored in a *build* file, which in turn is used to create an appset. The appset contains all the files, folders, and registry settings needed to reproduce the application on the client user's PC.

Variations among applications and installation can sometimes make publishing seem complex. For example, many applications integrate themselves deeply into the Windows operating system environment. Software vendors also use different installation programs (such as InstallShield and InstallAnywhere). Moreover, some applications integrate with other applications.

To reduce this complexity, this guide presents a best-practice approach to help ensure smooth and successful publishing. Figure 1 shows the overall publishing process. "Publishing Steps" is a step-by-step overview of the publishing process. "WSM Publisher File Types" describes the various file types used by WSM Publisher.

Figure 1 Workflow



Publishing Steps

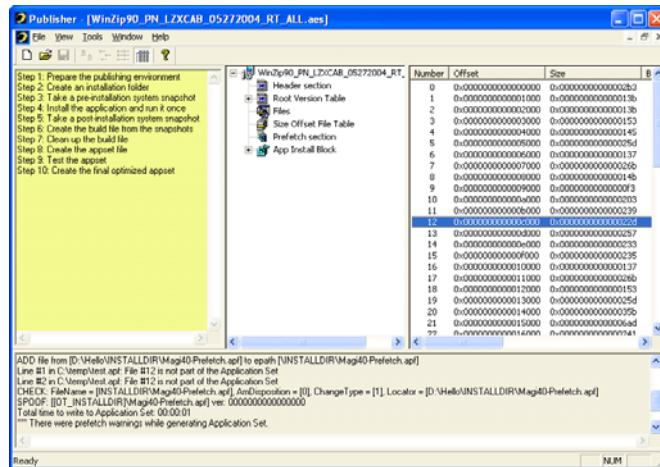
Each of the steps in this section describes one of the steps depicted in Figure 1.

 **Note**

The Project Wizard is available to guide you through some of the steps. It covers all of steps 2 through 6, and part of step 7 (the macro-based cleanup section). You also have the option of performing the steps individually.

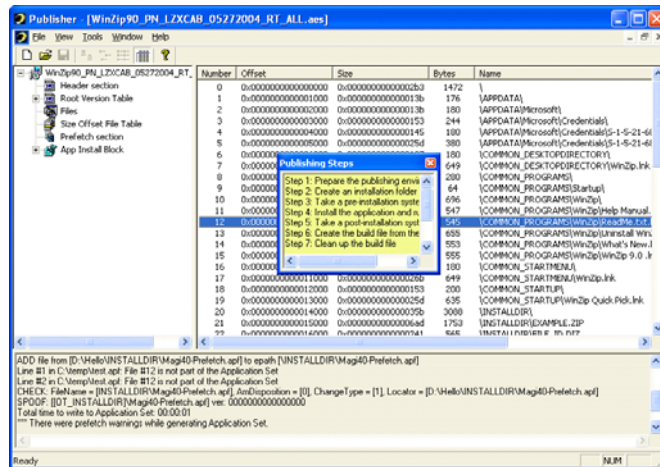
You can obtain a quick online reference of these steps by choosing Help | Publishing Steps (selecting the Publishing Steps command from the Help menu). A pane with a yellow background appears with the various steps listed, as shown in Figure 2:

Figure 2 Publishing steps - docked



If you prefer for these steps to be displayed in a floating window, click on one of the edges of the pane and drag it out to undock it. You can dock it back by clicking on the floating window's title bar and dragging it to one of the main window's edges.

Figure 3 Publishing steps - undocked



Step 1: Prepare the Publishing Environment

Preparation is critical to trouble-free publishing. A properly configured publishing environment is one in which the recommended hardware and operating system that are dedicated to publishing and can be imaged and restored easily. Each application to be published must be installed on a platform that has no other applications installed other than perhaps a ghosting application which is used for imaging and restoration. To reduce the artifacts that can be introduced into the system, you should always use a "clean" system for publishing. For more information on this step, refer to "Getting Started."

Step 2: Create an Installation Folder

After preparing the publishing environment, you create an installation folder for the target application. This folder must be created before you take the first snapshot of the system. For more information on this step, refer to "Creating Build Files."

Step 3: Take a Pre-installation System Snapshot

After the system is clean, you take the first snapshot. This snapshot of the system reflects the system state before the application is installed. This snapshot will be compared with a second snapshot of the system after the application has been installed. The differences between the snapshots are used to determine the files, configuration, and registry information that is needed to stream the application to the client. The first snapshot is saved as the pre-installation snapshot file. For more information on this step, refer to "Creating Build Files."

Step 4: Install the Application and Run it Once

After you create the first snapshot, you install the target application. Many application installers have a simple install mode (usually the default) and a custom install mode. To publish, however, it is best to use the custom install feature and install all the components. Because the components are all on-demand, it makes sense to publish everything possible for the target application. After you install the application, you run it to verify that all components are properly installed, and then you test all application menu options and features. For more information on this step, refer to "Creating Build Files."

Step 5: Take a Post-installation System Snapshot

After the target application has been installed and run, it is time for the post-installation snapshot. This snapshot is taken in exactly the same fashion as the first snapshot; however, the second snapshot shows the changes in the system due to the installation, which is all the files and configuration setup required for the target application. If target application or system changes are required, you would do them before the second snapshot is taken.

There is no limit to the number of snapshots that you can make. You can make changes at any time and take another snapshot. To publish a new application, however, you must set up a new environment in place of the existing one. For more information on this step, refer to "Creating Build Files."

Step 6: Create the Build File from the Snapshots

The pre-installation snapshot is compared to the post-installation snapshot file to determine what has changed on the system after the target application was installed. This difference ideally contains only the target application. However, during the installation it is possible to pick up unwanted artifacts.

The output from the comparison of the snapshots is the *build file*. The build file contains all of the files and Windows registry entry differences between the two system states. Most if not all of the files and entries are specific to the target application; however, sometimes artifacts are introduced into the process. *Artifacts* are any system information not related to the target application. For example, if you browsed the Internet after installing the target application, temporary files would be created on your system and would likely appear in the build file. Some artifacts can cause problems with the published application. For more information on this step, refer to "Creating Build Files."

Step 7: Clean-up the Build File

The most involved part of publishing is cleaning the build file and identifying and removing artifacts. This may seem complex at first, but requires only a simple understanding of how the Windows file system and registry work. After you have published a few applications, you will learn to quickly recognize patterns in the build file and identify and remove artifacts.

Keep in mind that cleaning up the build file can be an iterative process that requires testing between each successive build file. You can open the build file as necessary to modify it.

As a learning aid, two case studies have been provided for your review. Macros are available to speed up the clean-up process. Macros contain pre-recorded actions for cleaning the build file. In fact, in many cases the macros supplied with WSM Publisher are adequate to clean the build file. For more information on this step, refer to "Modifying Build Files."

Step 8: Create the Appset File

After the build file is clean, you can generate the application set, called the *appset*. During this process, you can specify the target operating systems. Several options are available for optimizing the appset (such as compression, encryption, and the inclusion of a prefetch file when recreating the appset for republishing), but you can skip these the first time around and then come back to them after testing the appset to optimize it. For more information on this step, refer to "Generating Appsets."

Step 9: Test the Appset

The appset is loaded onto a server for testing. Testing does not need to be rigorous functional testing, but rather a broad test of the standard features in the application. For example, all menu options should be tested to verify that all parts of the application are installed and captured. If the application does not work as expected, some part of the target application installation or cleaning of the build file was not completed. You need to go back to the build file and perform additional cleaning or modifications, generate a new appset, and test again. For more information on this step, refer to "Testing and Optimizing."

Step 10: Create the Final Optimized Appset

After you determine that the appset is satisfactory and the application functions as intended, you can optimize the appset with compression, encryption, and the inclusion of a specific prefetch file when recreating the appset for republishing. Once optimized, the appset is complete and can be loaded on the WSM servers for user subscriptions. For more information on this step, refer to "Testing and Optimizing."

WSM Publisher File Types

WSM Publisher uses various types of files that are briefly described in Table 1 along with their associated file extensions. Double-clicking on any of these files from within Windows Explorer opens the file in WSM Publisher or an appropriate editor (depending on the file type). More details on each file type are given in the sections where the file type is primarily used.

Table 1 WSM Publisher File Types

Ext.	Description	Double-click action
.aeb	WSM Publisher Build Contains a list of files and registry entries changes that are needed to generate an appset.	Opens build in WSM Publisher
.aec	WSM Publisher Configuration Contains configuration information used to build an appset (for example, installation folder, target OS, etc.).	Opens configuration file in Notepad
.aem	WSM Publisher Macro Stores recorded actions that can be replayed on an application build.	Opens macro in WordPad
.aep	WSM Publisher Project Keeps track of state and configuration information used by the Project Wizard.	Opens project in WSM Publisher
.aes	WSM Application Set Contains a repackaged version of an application that is ready for on-demand delivery to WSM clients via a WSM server.	Opens appset in WSM Publisher
.apm	WSM Publisher Macro Log Keeps a running log of all actions performed during the editing of an application build.	Opens macro in WordPad
.ss	WSM Publisher Snapshot Stores an image of the current state of a machine's files and registry entries.	Opens WSM Publisher (however, the contents are not viewable)



3

Getting Started

This chapter contains hardware and software requirements and the procedures you must complete to install the WSM Publisher.

Hardware and Software Requirements

Normally, you should publish an application on the same OS as the OS used on the user's target desktop. In addition to WSM Publisher requirements, each machine must meet or exceed the system requirements of the application being published. The following requirements have been established based on acceptable compression and encryption performance.

- Operating System: Windows XP or later
- CPU: 1.5 GHz CPU or higher
- RAM: 512 MB
- Disk Space: 40 GB

About Publishing Performance

The publishing process is both I/O- and CPU-intensive, and performance measurements have shown that the following criteria, in decreasing order, have the greatest effect on publishing speed:

- Publishing on a physical machine instead of a virtual machine (>30% speed improvement*)
- SCSI instead of IDE hard drives
- Faster front side bus
- Faster processor
- Dual processor instead of single processor (<10% speed improvement*)

* These measurements are provided only as a reference. Actual results may vary greatly based on your hardware and software system configuration.

Preparing a Dedicated Machine

It is recommended that you use a dedicated machine so that the Publisher does not notice any software, files, or registry information other than information for the Windows OS.

The dedicated machine should have all the latest OS service packs installed. It is assumed that the OS service packs fix any bugs in the OS but do not cause any application-specific files or registry settings to be copied into the system that could cause an application installer to take an erroneous path.

It is also recommended that you use a disk-imaging or ghosting application to establish a clean machine for a new application-publishing project. The disk image of a dedicated machine is saved as a read-only file (for example, WinXP_base.dsk).

Setting-up Disk Space for Publishing

To ensure adequate disk space, WSM Publisher requires a segmented drive space for the application to be published. You can install or rename drives via the Computer Management feature in Windows XP or later. To re-partition a drive, use a third-party application such as Partition Magic.

- Name the new drive space O: (letter O). Either install an additional hard drive or re-partition your current drive, or change the drive letter mapping.
- If there is only one partition (for example, C:), either install an additional hard drive or re-partition your current drive.
- If you have two or more partitions (for example, C: and D:) you must either rename one partition to "O:" or re-partition one of the drives.

Installing WSM Publisher

To install WSM Publisher:



Note

WSM Publisher will be installed in the Program Files folder on your C: drive.

1. Double-click on the installer file to open the Welcome window.
2. Click **Next**.
3. Read the license agreement, and select **I accept the terms in the License Agreement**.
4. Click **Next** to begin installation.
5. After the installation is complete, click **Finish**.



Note

To launch WSM Publisher, click **Start | Programs | WSM Publisher**, and select **WSM Publisher**.

4

Creating Build Files

This chapter provides the detailed procedures you need to create a build file. You can create a build file by using Project Wizard or you can create a build file manually.

As described in "Publishing Overview," creating a build file is a multi-step process. Project Wizard is available to guide you through the various steps and automatically set some configuration parameters for you. Although you can Project Wizard to create new build files, you can also manually create build files in cases where more configuration flexibility is needed.

Creating a Build File Using Project Wizard

This section shows how to use Project Wizard to create a new build file. Although this is the preferred approach for creating build files, it is recommended that you read "Creating a Build File Manually" to better understand the build creation process.

About a Project

Creating a build involves creating various files: two snapshot files, a configuration file, and the build file itself (see "WSM Publisher File Types"). Other files are associated with the process as well as the installer for the target application and the macros to be executed on the build file after it has been created.

When you use Project Wizard to create a build file, an additional project file with the extension .aep gets created as well. The .aep file is what glues the various other files together. It also allows Project Wizard to automatically configure some of the parameters for you. For example, so that you don't have to specify a name for each file yourself, the .aep file automatically names some of the files listed above based on the name you provide for the project. The .aep file also allows Project Wizard to resume the build creation process if the process is interrupted for any reason (for example, if a system reboot is required after installation of the target application).

Step 1: Launching Project Wizard

To launch Project Wizard:

1. From the Tools menu, select **Project Wizard**.
2. If you wish to create a new build, complete the following:
 - a. Select the Create a new project option.
 - b. Enter a name for your project under Project Name, usually the full name of the target application (for example, XYZ Software 6.0).



Note

The name you enter for the project will be used as the default name for many auxiliary files used to create the build, including the name of the build and project files themselves.

The Project Wizard will automatically populate the Project Location field based on the project's name with the following:

```
<My Documents>\My Packages\<project name>\<project name>.aep
```

where *<My Documents>* is your local My Documents folder and *<project name>* is the name you entered above.

**Note**

If you wish, you may enter a different path and file name.

3. If you want to resume working with an existing project, do the following:
 - a. Select the **Load an existing project** option.
 - b. Click **Browse**, locate the project file you want to open, and double-click it.
4. Click **Next** to open the Installation Information page.

**Note**

If you loaded an existing project, the next wizard page and information that you view will depend on how far along you got in the Project Wizard when you last had the project opened. For the remainder of this section, it will be assumed that you are creating a new project.

Step 2: Providing Installation Information

Now you provide information that Project Wizard will use to create an installation folder for you and launch the appropriate application installer later on:

1. In the Installer Location field, enter the complete command line for the target application's installer, including any applicable command line parameters (however, if the msi file path is given in this field, an error message will appear when **Launch Installer** is clicked).
2. Click **Generate GUID**. This creates a unique application ID number (UAID) to prevent installation folder name collisions between applications.

For more information on UAIDs, refer to "Step 1: Setting-up an Installation Folder."
3. Specify the drive where you want to install the target application under Installation folder.
4. If you want to name the installation folder for the target application using the UAID, select the **Using unique application ID** option. Otherwise, enter a name for the folder under Installation folder.
5. If you want to change the default settings used to take the pre-installation snapshot, click Advanced Snapshot Settings, make your changes, and click **OK**.

For more information on these settings, refer to "Step 2: Creating a Pre-installation Snapshot."
6. Click **Next** to open the Installer Launch page.

Step 3: Launching the Installer and Running the Application

At this point, Project Wizard is ready to launch the installer based on the command line entered earlier:

**Note**

If this is the first time you publish an application, refer to "Step 2: Creating a Pre-installation Snapshot" and "Step 3: Installing the Target Application" for important tips and recommendations.

1. Click **Launch Installer** and install the application as described in "Step 1: Launching Project Wizard."
2. When the installation is complete, you should run the application once, as described in "Step 4: Running the Application," and then return to Project Wizard and click **Next** to open the Record System Changes page and proceed with the build creation process.

**Note**

If the installer requires a system reboot then, after the system restarts, run the installed application once as indicated above, then launch WSM Publisher and open the project as shown in "Step 1: Launching Project Wizard" to resume the build creation process.

Step 4: Recording System Changes and Creating the Build

To record system changes and create the build:

**Note**

After this step, Project Wizard takes a post-installation snapshot of the system, compares it with the pre-installation snapshot, and creates a build file based on the changes that are found. Project Wizard will then automatically apply to it any macros that you may have specified.

1. Click **Edit Macros**.

The Play Macros dialog box appears.

2. Click **Include macros in default location**.

This step is recommended. It adds to the list all the macros located in the macros subfolder of the WSM Publisher installation folder. These macros include the generic macros provided with WSM Publisher as well as any macros you might have added to that subfolder. For more information about these macros, refer to "Using Macros for Automated Clean-up."

3. If you want to change the macros that should be applied to the build, follow these instructions:

- To add a macro to the list, click **Add Macro**, locate the macro you want to add, and double-click it.
- To remove a macro from the list, click it, and then click **Remove Macro**.
- To keep a macro in the list but not have it played, clear the check box next to it.
- To change the order in which macros are played, click a macro, and then click the up and down arrows to the right of the list (not the scroll bar arrows) to move the macro within the list.
- To view a description of a macro, click it in the list. If any comments are present at the top of the macro script, they are shown in the description field.

4. Click **OK**.
5. To change the settings for the post-installation snapshot, click **Advanced Snapshot Settings**, make your changes, and click **OK**.

**Note**

In general, you should use the same settings for both snapshots. So, if you change the settings for the pre-installation snapshot earlier, you should make the same changes for the post-installation snapshot.

6. Click **Next**.

Project Wizard takes a post-installation snapshot, compares it to the pre-installation snapshot taken earlier, and creates a build file based on the comparison results. It then applies the macros that you specified, if any. When Project Wizard is finished, the Summary page appears.

Step 5: Project Wizard Summary Page

The Summary page lists all the files that Project Wizard generated, including:

- Project file (.aep)
- Build files before and after macros are applied (.aeb)
- Configuration file (.aec)
- Pre-installation and post-installation snapshots files (.ss)

The Summary page also gives you the option to automatically launch the appset creation wizard.

Use the following guidelines:

- If you wish to perform additional modifications to the build before creating an appset, leave the **Launch Application Set Wizard** option cleared and click **Finish**.

**Note**

For information on what to do next, refer to "Modifying Build Files."

- If you wish to create an appset immediately without making any changes to the build beyond those already performed by Project Wizard, select the **Launch Application Set Wizard** option and click **Finish**. This will launch the Application Set Creation Wizard and go directly to the Main Options page with the UAID already pre-populated.

**Note**

For information on what to do next, refer to "Generating Appsets."

Creating a Build File Manually

If you decide not to use the Project Wizard to create a build file, you can perform the individual steps manually. Creating a build file manually entails completing these steps:

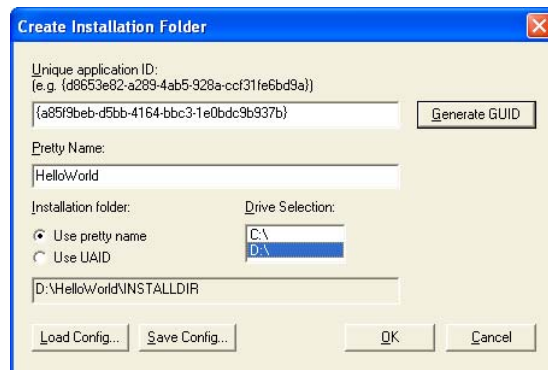
1. Setting-up an installation folder.
2. Creating a pre-installation system snapshot.
3. Installing the target application.
4. Running the target application.
5. Creating a post-installation system snapshot.
6. Creating the build file from the snapshots.

Step 1: Setting-up an Installation Folder

The first step is to set up a folder in which to install the target application:

1. On the File menu, click **Create Installation Folder**.

Figure 4 Create Installation Folder



2. If you are republishing an application and want to use an existing configuration file, click **Load Config**, select the configuration file, and then click **Open**.

The fields are populated from the information stored in the configuration file. At this point, you can continue with step 5.

 **Caution**

Only use a saved configuration file to republish the same application for which it was originally created, and on the same OS as originally published. Do not use a saved configuration file to republish on a different OS or to publish a different version of the application.

3. If you are publishing the application for the first time, click **Generate GUID**. This creates a unique application ID number (UAID) to prevent installation folder name collisions between applications.

 **Note**

As the Unique application ID field fills in and other options are set, the Installation folder field is updated with the path of the installation folder to be created.

The Unique application ID appears as the Application ID on the WSM server's Add Application and Edit Application Web pages.

Although not recommended, you can manually enter the GUID according to the format {xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}; where x is a hexadecimal digit from 0 through 9 or a through f.

4. If you want to give the installation folder a user-readable name (such as the default name used by the application's installer) rather than a GUID, specify a user-readable name in the Pretty Name field and select the **Use pretty name** option.

**Note**

You may specify a pretty name even if you select the Use UAID option. It would then get stored as part of the appset configuration although it would remain unused. Similarly, you may specify a UAID even if you select the Use pretty name option. This information will be used for the client mounting name when you create an appset.

5. To use an installation drive other than the default, select a different drive in the Drive Selection list.

**Note**

If the O drive is available as a local non-removable drive, it is automatically selected. Otherwise, the first available local non-removable drive is selected.

6. Click **Save Config** to save the settings to an appset Configuration file (with an .aec extension).

The Publisher uses the configuration file to build an appset. You can save the configuration file in any folder.

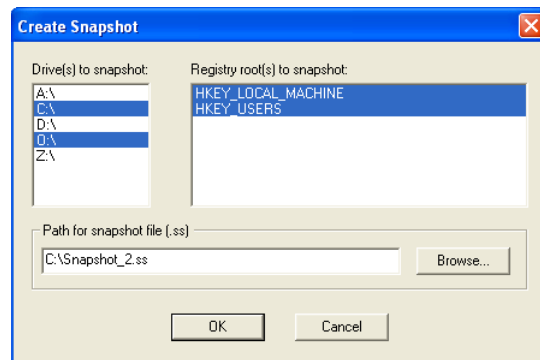
7. Click **OK** to create the installation folder and close the dialog box.

The folder is created and ready for the application to be installed.

Step 2: Creating a Pre-installation Snapshot

You need to capture an image or snapshot of your system configuration (files, folders, and registry entries) before installing the application. WSM Publisher compares the pre-installation snapshot with the snapshot taken after installation to identify changes to the file system and registry and to record the changes into a build file. Make sure you have everything ready before the installation to prevent artifacts during installation.

1. Insert the application CD in the drive.
 - If there is adequate space on the hard drive, copy all the files from the installation CD to the O: partitioned drive (create a folder as needed) and then remove the CD.
 - If there is not adequate space on the hard drive, leave the CD in the drive.
2. On the File menu, select **Create Snapshot**.

Figure 5 Create Snapshot - pre-installation

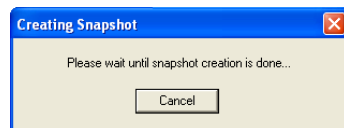
3. In the Drive(s) to snapshot list, select both the **C:** and **O:** drives.

Leave all items in the Registry root(s) to snapshot list selected (the default). These are registry root keys that need to be archived before modifications are made by the application installation.

4. In the Path for snapshot file (.ss) field, enter the snapshot path and file name (for example, C:\Snapshot_1.ss). The file can be saved under My Documents or any folder of your choice (click **Browse** to navigate to the folder).

Tip: Give the file a name to distinguish it from the subsequent snapshots you will take. For example: SnapshotFirst.ss

5. Click **OK** to create the snapshot.

Figure 6 Creating Snapshot

6. Click **OK** when the snapshot is completed.

**Note**

Avoiding Artifacts - Artifacts are unwanted changes or additions that can be introduced between the first and second snapshots and which can end up in the registry and file system. Use the following guidelines to avoid or reduce artifacts.

Avoid running applications or executing commands on the publishing machine between the snapshots.

Do a dry run of the installation on a machine other than the one used for publishing so that you know what to expect.

In general, minimize all machine activity after the first snapshot is taken.

Step 3: Installing the Target Application

Follow the instructions provided with the application to install it. Install the new application on the O: drive in the folder created in "Step 1: Setting-up an Installation Folder" (for example, O:\{GUID}\INSTALLDIR). During the installation, do the following:

1. Copy the vendor's original license text and save it to a text file. The file can be used as the application license file for creating the appset (see "Generating Appsets").
2. If you are prompted for any user or company-specific information, enter generic identification information. For example, you can enter "User" or "Employee" for the user name. For company name, you can enter the name of your company or the name of your customer's company.
3. Make a note about whether installation requires a reboot. You will also need to specify this for appset creation.



Note

Handling Machine Reboot Requests - Generally, you can ignore machine reboot requests during installation of an application. All relevant information that the Publisher needs will already be in effect by the time the second snapshot is taken. If the installation requires a reboot, the system creates artifacts that will be caught in the second snapshot. (for example, new hardware or drivers). Be sure to exclude new artifacts that are found after the reboot.

Step 4: Running the Application

Before creating the post-system installation image, you must run and use the application at least once to ensure that all necessary files, registry entries, configuration, license compliance and ActiveX controls registration are captured. If the entire application is not loaded, end users may get prompted to insert CDs. Some entries are not made until the application is run for the first time.

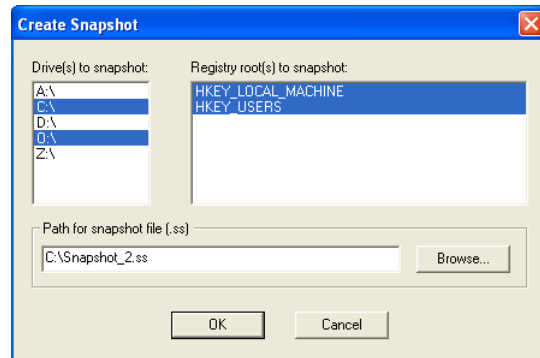
Use the following guidelines to run the application and verify that the entire application is loaded on the publishing machine:

- Start the application
- Click and use all of the menu items
- Click and use all of the toolbar items
- Perform a typical task and usage scenario for the application.
- Close the application.

Step 5: Creating a Post-installation System Snapshot

Use the following procedures to capture an image of your system configuration after you have installed the application and run it once. This procedure is identical to capturing the first snapshot.

1. On the File menu, select **Create Snapshot** to open the Create Snapshot dialog box.

Figure 7 Create Snapshot - post-installation

2. In the Path for snapshot file (.ss) field, enter the snapshot path and file name (for example, C:\Snapshot_2.ss).
3. Click **OK** to start the snapshot process.
4. After the Publisher finishes processing the snapshot and displays the Done Creating Snapshot message, click **OK**.

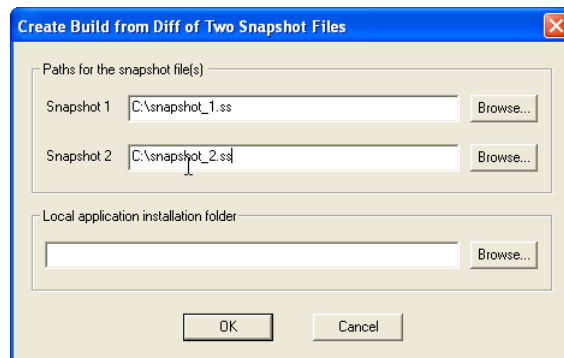
**Note**

The post-installation snapshot may take several minutes. It should take longer than the pre-installation snapshot, depending on the amount of system changes performed by the application installation and on system speed.

Step 6: Creating the Build File from the Snapshots

The next requirement is to generate a list of the changes made to the system during the installation and initial execution of the application. These changes are then saved to a WSM Build (AEB) file with the .aeb extension. An AEB file contains file and registry information that is used to generate a build file and eventually an appset.

1. On the File menu, select **New Build**, and then select **From Diff of Two Snapshots**.

Figure 8 Create Build from Diff of Two Snapshot Files

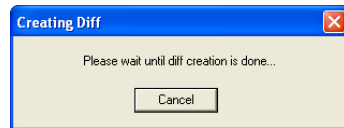
2. In the Path for the snapshot file(s) area, locate the pre-installation and post-installation snapshot files by clicking **Browse** next to the Snapshot 1 and Snapshot 2 fields, respectively, or manually enter the paths.

**Caution**

When entering the location paths, do not type a trailing backslash (\).

3. In the Local application installation folder area, click **Browse** to navigate to the folder on the O: drive where the application has been installed (for example, O:\{GUID}\INSTALLDIR), as indicated in "Step 3: Installing the Target Application").
4. Click **OK** to start the comparison of the snapshot images.

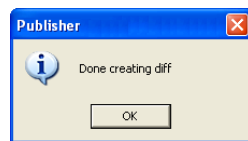
Figure 9 Creating Diff



Note

Depending on the extent of system changes, the comparison may take several minutes to complete.

Figure 10 Done Creating Diff



5. Click **OK**.

WSM Publisher now displays the files, folders, and registry entries that have changed during application installation in the left and right panes of the Publisher window, as described in "About the Publisher Build File Window."

Saving the Build File

To save the Build file:

1. From the File menu, select **Save Build As**, and enter a name for the build file where you wish to save the recorded, unmodified, changes (for example, MyApp_unmodified.aeb).
2. Click **Save**.
3. Save a copy of the file as a backup original and unchanged file. By default, files will be saved in the folder you originally designated for the appset Configuration file in "Step 1: Setting-up an Installation Folder."
4. From this point on, save the changes as you proceed by choosing the **Save Build** command on the File menu or by pressing **Ctrl+S**.

After you have created a build file, you can clean up and fine-tune it using the procedures in "Modifying Build Files."



5

Modifying Build Files

This chapter contains the procedures you need to clean up and fine tune a build file.

The build file identifies the files and configuration settings needed to replicate the application on an end-user's system. Generally, not all of the files and registry entries in a new build file are required. Also, some of the differences captured between system snapshots may be artifacts not related to the application. For example, if after installing the target application but before taking the post-installation snapshot, you open Internet Explorer to check sport scores, this will update the Windows registry with the Web site name and update the Internet Explorer temporary files. This would result in artifacts being captured in the post-installation snapshot and, therefore, in the created build file.

To ensure an accurate image of the target application, you must clean up the build file by eliminating unwanted components and artifacts. You can use macros to perform common clean-up operations automatically, and then manually perform any clean-up not covered by the macros. At times, you may also have to insert files or registry entries into the build file (to restore items that you had previously removed, for instance).

In addition to clean-up operations, you can control how a particular file gets installed on an end-user's system and how it appears to the user. You can also specify what particular executables should be used for application usage tracking, and protect specific files from being copied, deleted, and overwritten.

For example, the following is an overview of the typical build file modification process (these steps are described in more detail in the following sections):

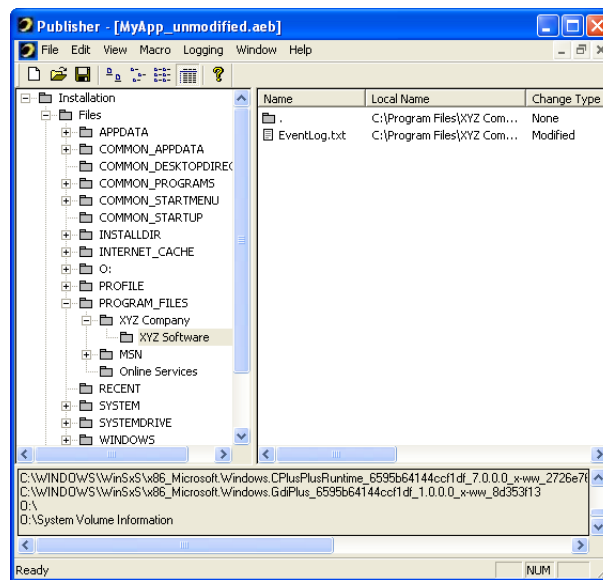
1. Perform automated clean-up:
 - Run generic macros.
 - Run application-specific macros (if available).
2. Perform manual clean-up:
 - Manually insert or remove files and folders.
 - Manually insert, remove, or modify registry keys and values.
3. Set access tokens.
4. Set file dispositions (as needed).

About the Publisher Build File Window

In the left and right panes of the Publisher window, WSM Publisher displays the files, folders, and registry entries that have changed during the installation of the target application. Affected folders and registry keys appear in the left pane, in a hierarchy under the Files and Registry Entries of the Installation tree, respectively.

Affected files and registry values appear in the right pane depending on the folder or registry key selected in the left pane, respectively.

Figure 11 Publisher Build File window



Undoing and Redoing Changes

WSM Publisher provides the capability to undo and redo build file changes. You can undo any number of consecutive changes starting with the last change by simply specifying how far back you want to go. You can also redo recently undone changes if necessary, as long as no other change has been performed after the last undo operation.

If a series of build file operations are performed by running a macro (as discussed later in this section), the entire sequence of operations is treated as if it were a single change for undo and redo purposes. You don't have to worry about tracking which operations were performed manually and which were performed using a macro.

WSM Publisher logs all operations that it performs as part of undoing or redoing a change into the log window, as for any other build file operation.



Note

Undoing or redoing a given change may require several operations. For example, if you remove a file to which you had associated an access token, undoing the operation would require adding the file back and setting an access token for it. This can result in multiple entries being written to the log window for a single undo or redo operation.

Undoing the Last Change

To undo the last change, click **Edit** to open the Edit menu and select **Undo** *<last change>* (where *<last change>* is a description of the last operation performed).

Undoing Multiple Changes

To undo multiple changes:

1. Select **Edit** to open the Edit menu, and select **Undo Multiple Changes**. The Undo Multiple Changes dialog appears.
2. Select the sequence of changes that you want to undo by clicking on the oldest change in the sequence. Doing so highlights the change and all the ones above it (that is, the ones that are more recent).
3. Click **OK**.

Redoing the Last Undone Change

To Redo the Last Undone Change, click **Edit** to open the Edit menu and select **Redo** *<last change>* (where *<last change>* is a description of the last operation performed).



Note

If you perform a build file change after the last undo operation, you will not be able to redo the undone changes.

Redoing Multiple Undone Changes

To redo multiple undone changes:

1. Select **Edit** to open the Edit menu and select **Redo Multiple Changes**. The Redo Multiple Changes dialog appears.
2. Select the sequence of undone changes that you wish to redo by clicking on the most recent change that was undone. This will highlight the change and all the changes above it (that is, the older changes).
3. Click **OK**.



Note

If you perform a build file change after the last undo operation, you will not be able to redo the undone changes.

About Cleaning Up Build Files

The build file clean-up process consists of first using macros to perform common clean-up operations automatically (see "Using Macros for Automated Clean-up"), and then manually doing any clean-up not covered by the macros (see "Cleaning-up the Build File Manually"). You can also create and save your own macros for current and future use.

Using Macros for Automated Clean-up

Macros are scripts of pre-recorded actions that can be "played" to modify a build file in an automated fashion. They save time and improve reliability. You can use Publisher macros to clean up a build file created under, and targeted for, a supported Windows platform.

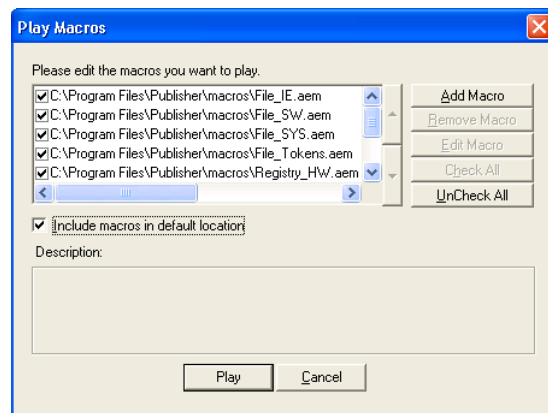
Not all clean-up operations apply to all applications. Some clean-up actions are general enough, however, and should be applied to all applications. WSM Publisher provides a set of generic macros that contain clean-up operations recommended for all applications. These macros are located in the macros subfolder of the Publisher installation folder and have the .aem file extension. Refer to Table 2 for a list of these macros and their descriptions.

In addition, you can use macros that are customized to specific applications.

To run macros:

1. From the Macro menu, select **Play Macro**.

Figure 12 Play Macros



2. Select the **Include macros in default location** option. This adds to the list all the macros provided with WSM Publisher. These macros are described in Table 2.



Note

These macros are located in the macros subfolder of the WSM Publisher installation folder. If you have saved your own macros in that subfolder, they will be added to the list as well.

Table 2 Generic Publisher Macros

Macro	Description
File_SW	Removes files and folders related to WSM Publisher
File_IE	Removes files and folders related to Internet Explorer (that is, cookies, Internet settings)
File_SYS	Removes system files and folders specific to the publishing machine
File_Tokens	Sets access tokens for all .exe files
Registry_SW	Removes WSM Publisher-related subkeys in the <i>Installer</i> key tree
Registry_HW	Removes all hardware keys specific to the publishing machine
Registry_MSI	Removes MSI-related keys
Registry_Sourcelist	Removes the Installation Source information
Registry_SYS	Removes System Settings keys specific to the publishing machine
CustomAppEvent	Allows for custom actions to be added to the Application Image.

3. Clear the check box to the left of the macros that you do not want to run. You can also click **Check All** or **UnCheck All** to select or clear the check boxes for all listed macros.

If you wish to add more macros to the list, do the following:

- a. Click **Add Macro**.
- b. Navigate to where the desired macro file is located and double-click it. This adds the macro to the list and its check box is selected automatically.
- c. Repeat these steps for each macro you want to add to the list.

**Note**

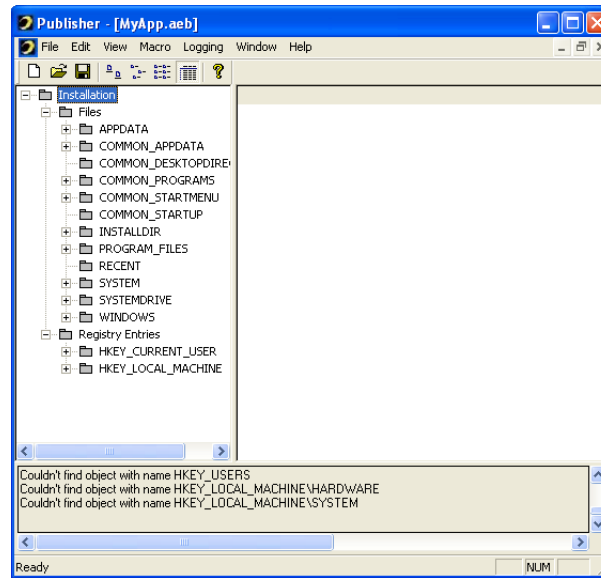
If a macro contains descriptive comments at the beginning of the file, clicking it displays these comments in the Description field.

If you do not have any custom actions, uncheck the CustomAppEvent macro

4. To change the order in which the macros will execute, select a macro and click the up and down arrows to the right of the list (not the scroll bar arrows) to move it up or down the list. Macros are executed in order from top to bottom.

5. Click **Play**.

All the macros you selected run in sequence, and their effect will be reflected in the left and right panes of the Publisher window, depending on whether the affected entries are visible in the window. Additionally, the performed operations are logged into the bottom pane just as they would be had they been performed manually.

Figure 13 Publisher

6. From the File menu, select **Save Build** to save the changes to the build file.

**Note**

For additional information on creating or modifying macros, refer to "Recording Macros" and "Publisher Macros."

7. Continue with "Cleaning-up the Build File Manually" to complete the clean-up process by manually modifying the build file.

Cleaning-up the Build File Manually

You should perform a manual clean-up of any unwanted artifacts not removed by the available macros. You can record the manual changes in your own macros, which you can reuse if the application needs to be republished. For the procedure on recording a macro, refer to "Recording Macros."

Build Tree Structure

As mentioned previously, the build file is represented in the left pane as a hierarchical tree called the build tree. The Installation item is at the root of the build tree and contains the Files and Registry Entries sub-trees, as shown in Figure 13.

Files Sub-tree

The Files sub-tree contains all the file system changes that occurred during the installation of the application being published. Directly underneath it are the symbolic representations of the installation folder created in the previous section and the Windows special folders affected by the installation.

Special folders are pre-defined system folders used for specific purposes (for example, My Documents and Program Files). Their exact paths on the client system depend on how the operating system is configured, so the Publisher refers to them using symbolic names that are system-independent. Files that are placed in these special folders are placed in the client machine's default locations when the application is activated.

**Note**

The Publisher uses the same symbolic representation for special folders as the Microsoft Windows Shell SDK, but with the CSIDL_ prefix removed. To find information on a particular special folder, you can do a search on its name at Microsoft's MSDN Library Web site (<http://msdn.microsoft.com/library/>).

Registry Entries Sub-tree

The Registry Entries sub-tree contains all the registry changes that occurred during installation of the application being published. Directly underneath it are the pertinent registry roots (or hives) affected by the installation.

Registry roots are keys in the Windows registry that are the roots of predefined top-level sub-trees (for example, HKEY_LOCAL_MACHINE and HKEY_CURRENT_USER). Because their names do not vary from one system to another, the Publisher refers to them by their standard names.

Clean-up Operations Overview

As mentioned earlier, the Files and Registry Entries sub-trees contain all file system and registry changes that occurred during installation. Items that were not captured in the build file, either because they have not changed during installation or because they were created later, can be manually inserted into the build file for inclusion in the appset. Existing items can be removed so that they are not included in the appset. You can insert them again later, if necessary.

The Publisher also allows you to exclude an item such that the item remains in the appset but is never streamed down to the WSM client. This may be a better alternative than removing the item if you think you may want to include it in the appset again in a future publishing. This option is mostly used to test if the build file will work without the excluded item before permanently removing it from the final appset. You can easily include previously excluded items to put them back into the build because they remain in the build tree, unlike removed items that get permanently taken out of the build tree.

Additionally, the Publisher allows you to easily inspect and modify registry values and value types.

The following sections provide general guidelines on what clean-up to perform and detailed steps on how to do the various operations.

Clean-up Guidelines

Choosing which files and registry entries to remove or modify requires some time and analysis. You will find that this process becomes faster and easier with practice. Review the following guidelines to help you decide what can be removed:

- Folders such as Cookies, History, Internet Explorer, and Publishing files (snapshot files) can be removed.
- Hardware entries in the key HKEY_LOCAL_MACHINE registry key should be removed. These entries can cause unpredictable results on a subscriber's machine.
- Empty keys that indicate modified under the Change Type column can be removed. For more information on change types, refer to "Modifying Registry Values and Types."
- Remove any shared .dll folder to avoid superseding the .dll folder of the subscriber's machine.

- For .ini files, check what the installation does by opening the file and looking for changes. If entries were added, it is better to implement them using an AppEvent DLL (for more information on AppEvent DLLs, refer to "About AppEvent DLLs.")
- Installer and CurrentVersion registry keys in HKEY_LOCAL_MACHINE should be removed.

The next section provides a list of valuable resources to help you learn more about items you are likely to find in a build file.

Where to find Additional Help and Information

If you are new to publishing, the clean-up process may seem complex at first. Although you are not expected to learn the purpose of each file, folder, and registry entry in a build, you will quickly become familiar with the process and which items to look for in a build. There are also various resources that can help in learning about the process and items you are likely to encounter in a build file.

Use the following guidelines for help and information:

- A good resource to learn what items to look for are the generic macros included with the Publisher. These commented macros describe the functions of many files, folders and registry entries, and what can happen if the items are left in a build file. Keep in mind, however, that there may be items not covered.
- The case studies at the end of this manual are an excellent hands-on learning tool.
- Another good resource for registry information is the Microsoft Knowledge Base (visit <http://www.wyse.com/support>). Do a search on a particular registry entry and you will most likely get information about it.
- You can also search on the Web for the registry entry. There may be more information about the entry or it may even be a discussion topic in a technology message board. Research and experience is the best way to get familiar with the Windows registry keys and values.
- Several publications dedicated to the Windows registry are also available and offer more detailed descriptions of the registry.

Inserting, Removing, Including and Excluding Entries

As mentioned earlier, items can be inserted into the build, removed from the build, excluded from the build, or put back into the build. These operations are performed using context menus.



Note

Inserting or including entries are collectively referred to as add operations, and removing or excluding entries are collectively referred to as prune operations.

Table 3 indicates which context menu to use for each operation. The command name matches the name of the operation to be performed. For example, to insert a registry key, right-click on the key under which the new key is located, and select **Insert Subkey**.

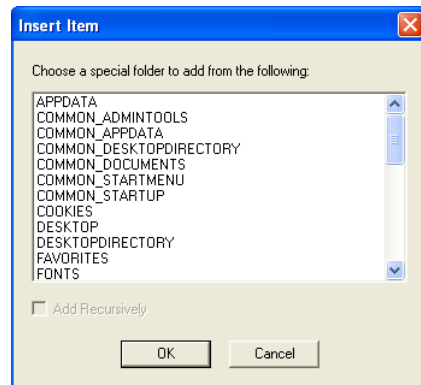
Table 3 Insert, Remove, Include, and Exclude operations

To...	This...	Right-click on this...
Insert	Special Folder	Files item
	Folder	Folder under which the folder to be inserted is located
	File	Folder under which the file to be inserted is located
	Registry Root	Registry Entries item
	Registry Key	Key under which the key to be inserted is located
	Registry Value	Key under which the value to be inserted is located
Remove	Any item	Item to be removed
Include	Any item	Item to be included
Exclude	Any item	Item to be excluded

To insert a special folder:

1. Right-click on the Files item in the build tree.
2. Select **Insert Special Folder**. to open the Insert Item dialog box with a list of all special folders not already in the build.

Figure 14 Insert Item



3. Select the special folder you wish to insert into the build, and click **OK**.



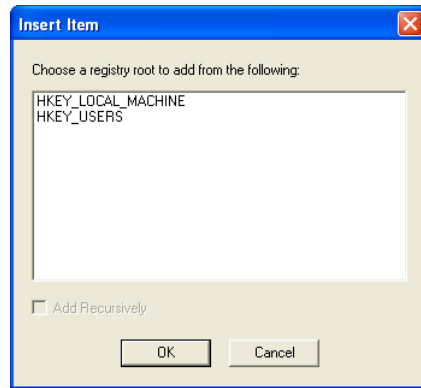
Note

As noted previously, the Publisher uses generic, system-independent names to refer to these folders because their exact locations may vary depending on operating system configuration.

Inserting a Registry Root

To insert a Registry Root:

1. Right-click on the Registry Entries item in the build tree.
2. Select **Insert Registry Root** to open the Insert Item dialog box with a list of relevant registry roots not already in the build.

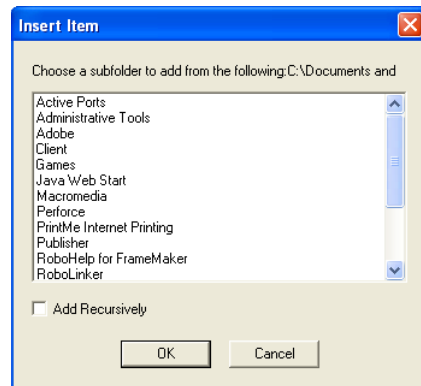
Figure 15 Insert Item - registry root

3. Select the registry root you wish to insert into the build, and click **OK**.

Inserting a File or Folder

To insert a file or folder:

1. In the left pane, right-click on the folder under which the item to be inserted is located.
2. Select **Insert File** or **Insert Subfolder** as appropriate to open the Insert Item dialog box with a list of relevant items not already in the build.

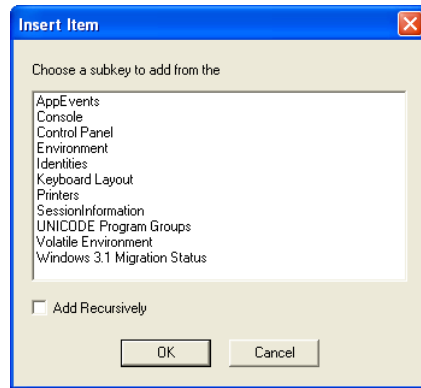
Figure 16 Insert Item - select subfolder

3. Select the item you wish to insert into the build.
4. If the item is a folder, you can select the **Add Recursively** check box to recursively insert all files and subfolders under the selected folder.
5. Click **OK** for the change to take effect.

Inserting a Registry Key or Value

To insert a registry key or value:

1. In the left pane, right-click on the key under which the item to be inserted is located.
2. Select **Insert Subkey** or **Insert Value**, as appropriate to open the Insert Item dialog box with a list of relevant items not already in the build.

Figure 17 Insert Item - select subkey

3. Select the item you wish to insert into the build.

If the item is a registry key, you can select the **Add Recursively** check box to recursively insert all subkeys and values under the selected key.

4. Click **OK** for the change to take effect.

Removing, Including, or Excluding an Item

To remove, include, or exclude an item:

1. Right-click the item (file, folder, registry key or registry value) that you want to remove, include, or exclude.
2. Select **Remove <item>**, **Include <item> In AppSet**, or **Exclude <item> From AppSet**, as appropriate.
3. Select **Yes** at the confirmation prompt for the change to take effect.

Generating, Adding, and Pruning Reports

It is sometimes useful to have a report of all entries that have been inserted or included into a build, or removed or excluded from it. WSM Publisher keeps track of these operations and is able to generate such reports on demand. You can also reset the tracking of these operations at any time (in which case tracking will start again).

Generating a Report of Included and Inserted Entries

To generate a report of included and inserted entries:

1. From the Logging menu, select **Create Adding Report**.
2. Specify the path of the text file where you wish to store the report, and click **OK**.

Generating a Report of Removed and Excluded entries

To generate a report of removed and excluded entries:

1. From the Logging menu, select **Create Pruning Report**.
2. Specify the path of the text file where you wish to store the report and click **OK**.

Resetting Tracking for a Report

To reset tracking for a report:

- To reset tracking for the Adding report, select **Reset Adding Report** from the Logging menu.
- To reset tracking for the Pruning report, select **Reset Pruning Report** from the Logging menu.

Modifying Registry Values and Types

You can modify the value and type of a registry setting to determine how the application functions on the client system. You can also configure a registry setting to take its value and type from another registry setting (such as a system setting, for instance) by setting its type to a special “copy” value.



Caution

Making incorrect modifications to the registry can cause the published application not to function properly or cause system instability.

Changing Registry Values

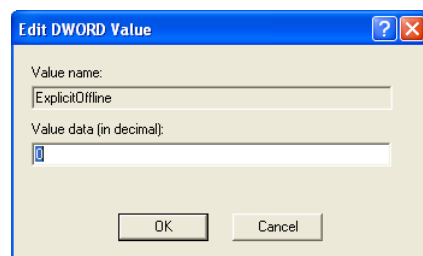
Registry values are data items that are stored in registry keys. They may represent configuration settings, references to file names, paths, or other registry entries, or even arbitrary data. You can select and modify a registry value with the Windows Registry Editor, but the advantage of using the Publisher is that it does not affect the registry on the publishing system, but only the build file information.

Such modifications are most useful to change user-specific information (such as the registered user name for the application) to make them more general. They can also be used to restore default configuration parameters such as the size and position of the application’s main windows.

To change a registry value:

1. Locate the registry value you want to modify.
2. Right-click the value and select **Modify Value** to open the Edit String Value dialog box.

Figure 18 Edit String Value



3. Enter the new value in the Value Data field and click **OK**.

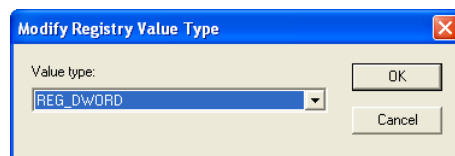
Changing Registry Value Types

Data is stored in various formats. Generally, the system uses a few simple formats, while applications, drivers, and so forth use more complex types defined for specific purposes. For example, REG_RESOURCE_LIST is a complex registry type used primarily by drivers. The Publisher allows you to change these value types, although for typical publishing such changes are unnecessary.

To change a registry value type:

1. Locate the registry value whose type you want to modify.
2. Right-click the value and select **Modify Value Type** to open the Modify Registry Value Type dialog box.

Figure 19 Modify Registry Value Type



3. Select the new type in the Value type list, and click **OK** to open the Edit String Value dialog box.
4. Update the registry value in the Value Data field, if necessary, and click **OK**.

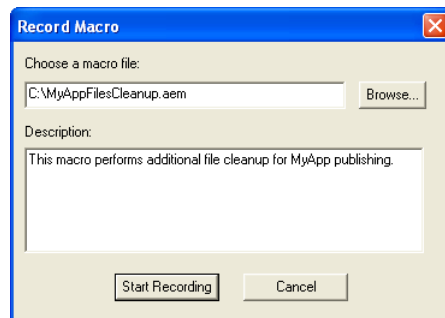
Recording Macros

If you clean the build file manually, you can record the manual operations into a macro for future use. You should record a macro whenever you want to record any changes made to the build file. A benefit of recording macros is that each macro serves as a reference to what has been changed in the file. For additional information on macros, refer to "Publisher Macros."

To record a macro:

1. On the Macro menu, select **Record Macro** to open the Record Macro dialog.

Figure 20 Record Macro



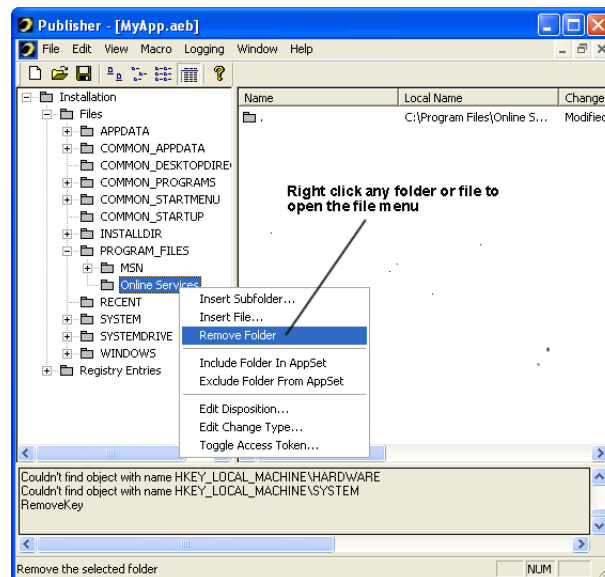
2. In the Choose a macro file text box, enter a path and file name for the macro (or click **Browse** to navigate to the folder where you wish to store the macro, enter a file name, and then click **Save**).

3. Enter a description in the Description field to explain the purpose of the macro. The description is optional but recommended.
4. Click **Start Recording**.

Actions that you perform in the left or right panes of the Publisher window are recorded in the macro, until you stop the recording.

Consider recording changes with a common purpose into separate macros. This provides flexibility if you need to republish an application later. You may also want to keep file changes separated from registry changes so that they can be applied independently. Figure 21 shows a folder being removed from the build file.

Figure 21 Remove Folder



5. To save the changes you have made, select **Save Build** on the File menu.

Recording Changes in a Macro Session Log

The Publisher can automatically record changes made to any .aeb file in every WSM Publisher sessions in a Macro Session Log file (.apm). This file has the same format as a regular macro (.aem) file, but it also contains comments indicating the start and end of every WSM Publisher session. The Macro Session Log file is intended for use as a running log of operations rather than a specifically targeted macro.

To record changes:

1. From the Macro menu option, select **Macro Log**.
2. Select the **Keep macro log of this session in file** option to enable session logging (this option is selected by default).
3. Specify the location and name of the log file you want to create. The file session.apm is used by default and it is located in the WSM Publisher installation folder.
4. If you want to append a comment to the log file, select **Insert Comment**, type in the text you want to insert, and click **OK**.
5. Click **OK**.

Setting Access Tokens

Access tokens are used to track usage on the client system and to apply security to files. With the access tokens enabled, the user cannot update, copy, or delete the file when the application is activated on the client machine. You can apply access tokens to executable files, libraries, or sensitive materials that should not be copied.

To apply access tokens on files, use the Toggle Access Token command and select from the following settings:

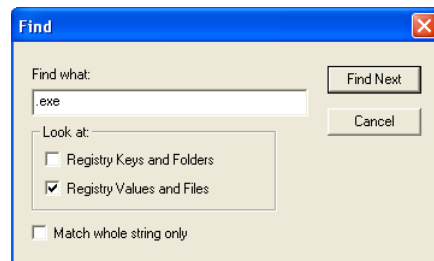
- Access token required - All selected files will have access tokens enabled.
- Based on name (.EXEs only) - By selecting access tokens required and based on names, this applies access tokens only to executable files (.exe) that are in the set of selected files.

You can easily check whether an access token has been set for a file by looking in the right pane under the Access Token Requested column.

To set access tokens:

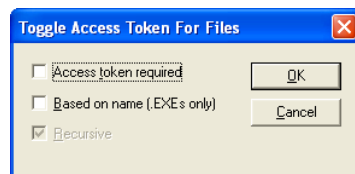
1. Locate a file for which an access token should be set. Follow these steps to use the Publisher's Find feature to locate the file:
 - a. From the Edit menu, select **Find**.
 - b. In the Find field, enter the appropriate file extension (for example, ".exe").

Figure 22 Find



- c. Clear the **Registry Keys and Folders** check box and make sure the **Registry Values and Files** is selected, and then click **Find Next**. From this point on, you can simply press the **F3** key to go to the next occurrence of the search string.
2. Right-click on the file and select **Toggle Access Token** (if you wish to set an access token for all files or executables under a given folder, then right-click on the folder instead and select **Toggle Access Token**) to open the Toggle Access Token For Files dialog box.

Figure 23 Toggle Access Token For Files



3. Select the **Access token required** check box.
4. If the item is a folder, you can select the **Recursive** check box to have the change apply recursively to all files and subfolders under the selected folder. In this case, you

should select the **Based on name (.EXE only)** check box if you want access tokens to be set only for .EXE files.

5. Click **OK** for the change to take effect.
6. Repeat the above steps for all files or folders for which an access token should be set.

**Note**

If you have not saved the build file since the last changes made to it, an error message is displayed. Save the file and try again.

If you have not set any access tokens, a warning is displayed and you are asked whether or not you wish to continue. In this case, select **No**, set appropriate access tokens, and try again.

7. On the File menu, select Save Build to save the changes you made to the build file.

Changing File and Folder Dispositions

The *disposition* of a file or folder refers to the status of the item and how it is handled in relation to WSM client. An item's initial disposition is set during installation based on how the item was created and captured in the snapshot. The Publisher allows you to change the disposition of an item to meet your requirements. Editing file dispositions can be used to arrange files when they are being activated on the client machine. It is also useful when the files need to be placed at a certain location on the client machine for an application.

File and folder dispositions are displayed in the right pane under the Disposition column and can be one of the following:

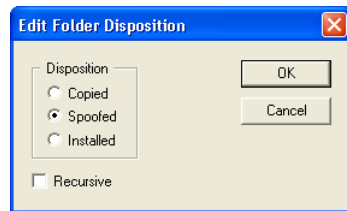
- **Copied** - The disposition is set to Copied to make the item immediately available to the WSM client. In case the Application Server is not available, the application will continue to run and function properly without fetching the files and folders from the server. For some applications, this solves a timing issue when loading different application components.
- **Spoofed** - The system sees a spoofed file or folder as if it were on the local drive even though the items are mapped to the WSM client's virtual network drive.
- **Installed** - Items marked as installed are available for streaming via the WSM client. The user will use these items on the WSM client's virtual network drive. This is the default disposition for items created during the installation process.

Editing a File or Folder Disposition

To edit a file or folder disposition:

1. Locate the file or folder with the disposition you want to edit.
2. Right-click the item and select **Edit Disposition** to open the Edit Folder Disposition dialog box.

Figure 24 Edit Folder Disposition



3. Select the Disposition option you want.
4. If the item is a folder, you can select the **Recursive** check box to have the change apply recursively to all files and subfolders under the selected folder.
5. Click **OK** for the change to take effect.

Viewing Change Types

The build stores a change type for each file, folder, and registry entry to indicate how a particular item was installed. An item's change type is displayed in the right pane under the Change Type column.

The change types are set by the Publisher during installation to be either Created if an item is new, or Modified if an existing item has been modified. You can use the change type as a reference to identify how a particular item was installed.

A change type can be one of the following:

- **Created** - Replaces the current file on the client system upon activation. The changes will be undone upon deactivation.
- **Modified** - Modifies the current file on the client system upon activation. The changes will be undone upon deactivation.

This page intentionally blank.

6

Generating Appsets

This chapter provides the detailed procedures you need to create an appset.

Creating an Appset

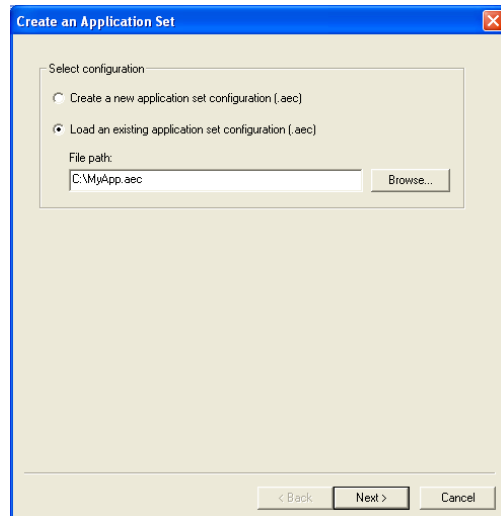
The appset packages all of the pieces of a Windows application in a format that can be streamed to the client. When creating the appset you select the operating systems for deployment and any options you want to optimize the appset (such as compression, encryption, and the inclusion of a prefetch file when recreating the appset for republishing). The completed appset is then loaded on the WSM server for testing and deployment.

Step 1: Loading the Configuration File

To load the configuration file:

1. On the File menu, select **Create Application Set** to open the Create an Application Set wizard.

Figure 25 Create an Application Set wizard



2. Select **Load an existing application set configuration (.aec)** to load the configuration file. This file was created when the application installation folder was created, as described in "Step 1: Setting-up an Installation Folder."
3. Various options in the appset creation wizard pages are populated automatically with the information contained in the Application Set Configuration file.
4. Click **Browse**, select the saved appset Configuration, and click **Open**.
5. Click **Next** to open the Main Options page.

Figure 26 Main Options

The Unique application ID field is populated with the UAID generated when the folder was set up. If you are republishing the same version of an application, use the same UAID.

6. In the Application set file path area, enter the path and name of the appset file you want to use, or click **Browse** to locate an existing appset to override.
7. In the Application set ID area, click **Generate GUID** to create the Application Set ID, which provides a version for the Application Server and the Publisher.

 **Caution**

When republishing an application, always generate a new ID for an appset, even if you intend to overwrite the old appset on the application server. Although you could use the same appset ID if you first remove the old appset from the application server, doing so would invalidate all subscriptions associated with the old appset! So, if your intent is to upgrade existing users to the new appset, use the upgrade process documented in the *Administrators Guide: Wyse WSM™* instead. One situation in which you may actually want to use the same appset ID is for testing purposes only, but never for a released appset.

 **Note**

The Application set ID appears as the File ID on the WSM server's Add Application and Edit Application Web pages. The GUID for the appset ID can be manually generated (not recommended) according to the format {xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}, where x is a hexadecimal digit from 0 through 9 or a through f.

8. Enter information in the Application name, Vendor name, and optional Application description on the server areas.

The Application name and Application description on the server entries are used as defaults for the Name and Description fields on the WSM server's Add Application Web page. Those, in turn, are used as the application's name and description on the subscription pages. The server does not use Vendor name.

9. The Client Mounting information specifies in which drive and folder the application will appear to be installed on a WSM client when the appset is activated. By default, these are populated with information stored in the appset configuration file, so it will match the installation folder path created earlier.

You can specify a different mount drive and folder from the mount drive and folder on which you actually installed the application on the publishing system because those are in fact independent.



Note

You must specify a UAID regardless of the mounting folder specified. The reason is that the mounting folder is used solely to represent the folder where the application is installed to the user, in the Windows UI. Internally, however, WSM always uses the UAID for the actual name of the folder. It just won't be visible to the user if a different mounting name is used.

10. Click **Next** to open the Authoring Information page.

Figure 27 Authoring Information

11. Enter your name in the Author name field.
12. (Optional) Enter comments in the Comments field.

You can use the Comments field to keep track of what is done in this version of the appset. You can retrieve the information with the `ess_dump.exe` utility that is provided with the WSM client.

13. Click **Next** to open the Supported Operating Systems page.

Step 2: Selecting Operating Systems

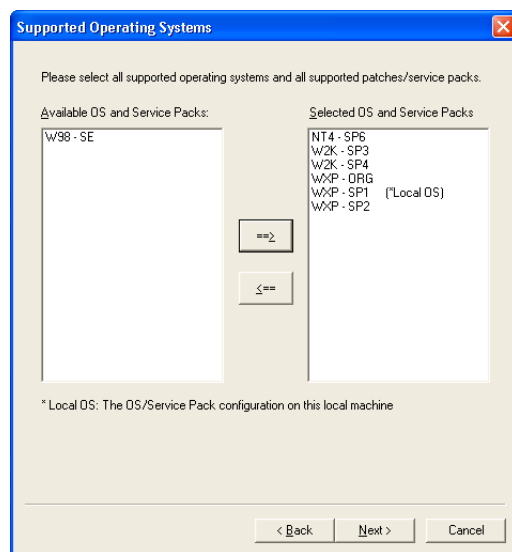
The low-level behavior of an application can vary greatly between different operating system (OS) and service pack (SP) combinations. Although you always create an appset on a particular platform (the local OS), you can mark it to run on multiple OS/SP combinations. For example, an appset published on a Windows XP SP3 machine will most likely run fine on a Windows XP SP4 system. However, an appset created on Windows XP SP4 may or may not run properly on Windows XP SP3 because of Windows XP-specific features or some low-level changes. For quality assurance, you should always test an appset on all platforms on which it is intended to run.

You can select the OS/SP combinations on which the appset will be allowed to run, or you can select the local OS only. After publishing an appset, you can easily change the list of supported OS/SP combinations, as described in "Modifying the List of Supported Operating Systems."

 **Caution**

Creating separate appsets for Unicode-based Operating Systems and non Unicode-based Operating Systems is recommended. If the OS/SP selection includes both Unicode-based and non Unicode-based Operating Systems, the appset may not function properly on the Unicode-based Operating Systems.

Figure 28 Supported Operating Systems



To select the operating system:

1. Select the OS/SP combinations on which the application will be allowed to run in the Available OS and Service Packs list, and then click ==>.

The selected items will be added to the Selected OS and Service Packs list.

The OS/SP running on the publishing system is identified in the list as Local OS for convenience.

 **Note**

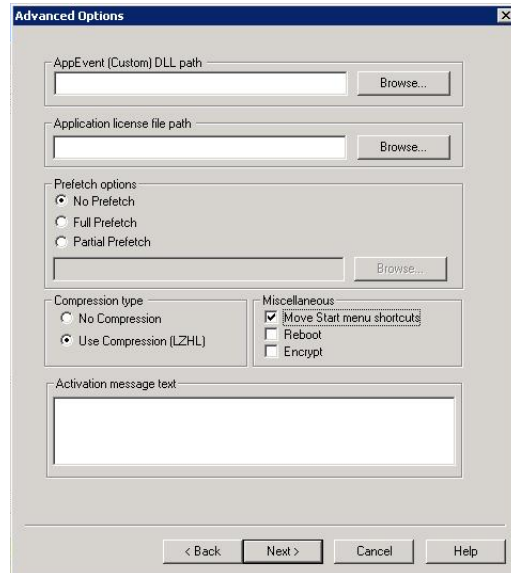
To select multiple consecutive items in the list, click the first item and then drag the mouse pointer to the last item. To select multiple non-consecutive ranges of items in the list, select the first range of items as described above, and then hold the Ctrl key down while selecting the remaining ranges of items. Holding down the Ctrl key preserve previous selections.

2. To remove OS/SP combinations, select them in the Selected OS and Service Packs list, and then click <==.
3. Click **Next** to open the Advanced Options page.

Step 3: Configuring Advanced Options

The Advanced Options page provides optional functions that you can ignore for the initial appset creation. However, information you need to configure these options is provided (for example, you can include a prefetch file when recreating an appset).

Figure 29 Advanced Options



Note

Selecting to include a prefetch file, compression, and miscellaneous options on the Advanced Options page are optional. After configuring advanced options, click **Next** to open the Summary page.

About AppEvent DLLs

To convert a locally installed application into a network streaming application, the Publisher takes a snapshot of the states of the installed application, and records the states in the appset file. When the Client activates an application, the snapshot is re-created on the user's system. However, sometimes the re-created static state of the application does not run the application. In these cases, you may need to adjust the states or the runtime environment before, during, or after activation.

To allow custom adjustments to the application on activation, the WSM Client provides an event-driven framework that can be used to trigger execution of custom code. This custom code is packaged as a dynamically linked library (DLL) and is inserted in the application appset during the publishing process. This DLL is called an *AppEvent DLL*. For more information on creating an AppEvent DLL, please contact "Wyse Technical Support."

Use the following guidelines:

- **AppEvent (Custom) DLL path** - If you have an AppEvent DLL file for the appset, specify it either by entering it manually or selecting it by using **Browse**. If you do not have a DLL at this time, ignore this field.
- **Application license file path** - If you have a license file for the application, specify it by either entering it manually or selecting it by using **Browse**. The contents of the file will then be displayed to the user when the application is activated. If you stored the application license text when installing the application, specify the path here.

About Prefetch Options:

You cannot use prefetching until the appset has been created and tested; therefore, if this is an initial creation, you can skip this option for now. Later (after initial appset creation), you can specify the inclusion of a prefetch file when recreating the appset for republishing. For information on creating a prefetch file and republishing an application with it, refer to "Creating and Using a Prefetch File."

Use the following guidelines:

- **Full Prefetch** - (*Recommended for WSM Site Deployments*) Select for high performance in a WSM Site deployment (as the appset at *Headquarters* is marked for prefetch, but the actual prefetching (streaming) of the whole appset to the client is performed at the *Linked Site*).
- **Partial Prefetch** - Partial prefetching is the process of streaming specific portions of an appset to the client as the appset is being streamed to the client machine for the first time (for example, at activation time). By streaming additional appset data in advance, prefetching results in a better response time when the user runs an application for the first time, or when the user uses common application functions for the first time. This can improve the user experience. To use prefetching, you specify the inclusion of a prefetch file when recreating the appset for republishing. This file describes which portions of the appset must be streamed to the client upon activation.

About Using Compression:

You can optimize the streaming of application data by compressing the appset. Selecting to use compression (select **Use Compression (LZHL)**) is optional when you create the appset. Note that using compression significantly increases the time required to generate an appset, especially when multiple publishing iterations are necessary. The best approach is to use compression only after the appset has been tested and is ready to be finalized. Then you can republish the application and apply compression.

About Using Miscellaneous Options:

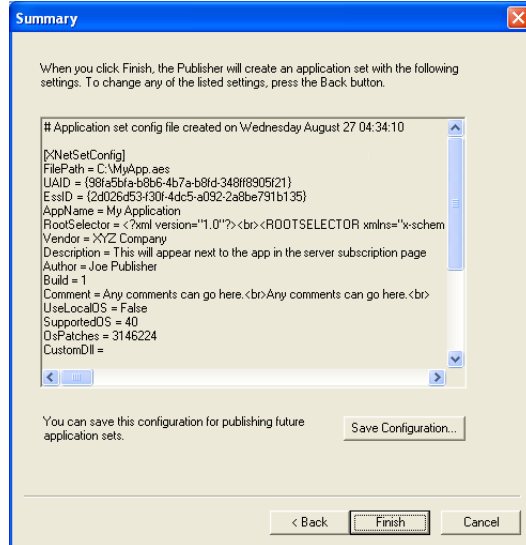
Use the following guidelines:

- **Move Start menu shortcuts** - Select to add the shortcuts of an application to the *Start* menu. This is required for shortcuts to work properly.
- **Reboot** - Select if the published application requires a reboot after installation (you noted this requirement when you installed the application).
- **Encrypt** - Select to encrypt the appset (note that this significantly slows streaming).
- **Activation message text** - If you want to display information upon activation of the application (for example, to prescribe specific usage instructions, and so on), enter the desired text.

Step 4: Saving and Creating the Appset

The Summary page provides a summary of settings for the application set.

Figure 30 Summary



1. To overwrite the Application Set Configuration file saved in an installation folder (as described in "Step 1: Setting-up an Installation Folder"), or to save the modified configuration as a separate file, click **Save Configuration**, select an existing configuration file or enter a new file name, and then click **Save** (if overwriting an existing file, click **Yes** when prompted for confirmation).

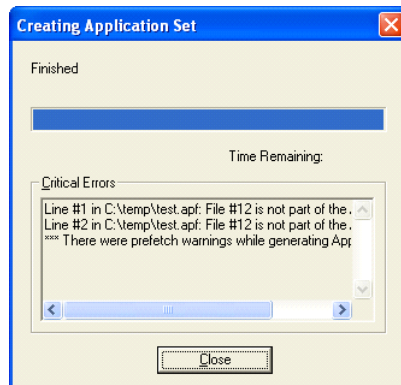
 **Caution**

The saved configuration file should be used only to republish the same application.

2. Click **Finish** to create the appset.

A dialog box appears showing a progress bar along with the elapsed time and the expected remaining time.

Figure 31 Creating Application Set



Appset creation may take from a few minutes to several hours depending on the complexity of the application being published, the speed of the system, and whether or not compression is used.

**Note**

If critical (but otherwise non-fatal) errors occur during the appset creation process, they will be reported in the Critical Errors field. The creation process will not be interrupted, however, as the resulting appset may still be functional. If you wish to interrupt the process, simply click Cancel.

3. Click **Close** when the Publisher finishes creating the appset.

All actions performed during the creation of the appset are logged in the bottom pane of the Publisher window along with the total elapsed time.

**Note**

If you get an error during when creating the appset, please refer to the WSM Publisher FAQ (accessible from the Help menu). If you do not find an entry that addresses the problem, please contact Technical Support.

Encrypting and Decrypting Appsets

WSM Publisher offers the capability to encrypt appsets using the Advanced Encryption Standard (AES) for securing appset content. However, since the encryption process may take several minutes depending on the size of the appset and the speed of the publishing system. It is a good idea to encrypt an appset during the final phase of publishing, after the appset has been thoroughly tested and before it is placed on the application server. You can also decrypt an appset from within WSM Publisher.

**Note**

If an encrypted appset is opened by any of the Publisher's functions, the Publisher automatically detects the encryption and provides the user with an option to decrypt the appset on the fly before opening it.

Encrypting an Appset

To encrypt an appset:

1. From the Tools menu, select **Encrypt Application Set**.

The Encrypt Application Set dialog will appear.

2. Specify the path of the appset that you wish to encrypt in the Input field.
3. Specify the path of the encrypted appset that you wish to create in the Output field.

**Caution**

If you specify the same file as in the Input field, the original (unencrypted) appset will be overwritten.

4. Click **OK** to initiate the encryption process.

Decrypting an Appset

To decrypt an appset:

1. From the Tools menu, select **Decrypt Application Set**.
The Decrypt Application Set dialog will appear.
2. Specify the path of the appset that you wish to decrypt in the Input field.
3. Specify the path of the decrypted appset that you wish to create in the Output field.



Caution

If you specify the same file as in the Input field, the original (encrypted) appset will be overwritten.

4. Click **OK** to initiate the decrypting process.

Viewing Appsets

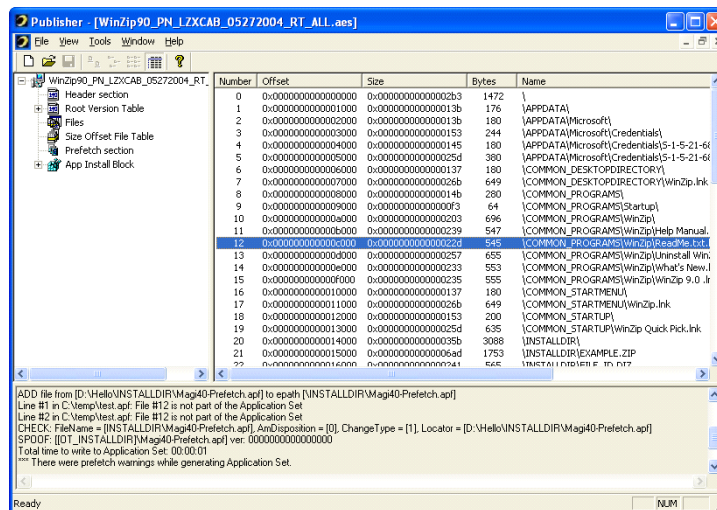
There may be times when you need to view the contents of an appset for troubleshooting purposes, or to extract a file from it. WSM Publisher allows you to open an appset and view its contents as described in the following steps.

Viewing the Contents of an Appset

To view the contents of an appset:

1. From the File menu, select **Open Application Set**.
2. Select the appset that you wish to view and click **Open**.
3. In the left pane, select the section of the appset that you wish to view, and its contents will be shown in the right pane.

Figure 32 Appset Viewer



Extracting a File from an Appset

To extract a file from an appset:

1. From the File menu, select **Open Application Set**.
2. Select the appset that you wish to view and click **Open**.
3. In the left pane, click on **Files**. The list of all files contained in the appset will appear in the right pane.
4. Right-click on the file you wish to extract and select **Extract File** from the menu.
5. Navigate to the location where you wish to save the file and enter a name for it or keep the default name.
6. Click **Save**.

What's Next

After creating a new appset, you can test and optimize it by using the procedures in "Testing and Optimizing."

7

Testing and Optimizing

This chapter contains the procedures you need to test and optimize an appset. It also contains information on correcting common problems that may occur with appsets.

After you have created an appset you can load it on an WSM server for testing and optimization. The appset is subscribed to from a client and then tested for complete delivery of contents and application functions. After eliminating any problems, and republishing if necessary, you can optimize the appset (for additional streaming efficiency) with compression, encryption, and the inclusion of a specific prefetch file when recreating the appset for republishing.



Note

It is recommended that you avoid testing on a publishing machine.

Loading Appsets

This section contains information on copying an appset to the depot folder on a server machine.

To load an appset:

1. Locate the AES file for the newly created appset file. The location was specified when you created the installation folder.
2. Locate the application server folder that was created during the installation of the Application Server.
3. Copy the AES file into the folder (for example, depot) created during installation of the application server.
4. Open a Web browser and go to the URL for the Application Server to open the Login page.
5. Log in with the administrator user ID and password established during the application server installation.
6. Use the procedures in the *Administrators Guide: Wyse WSM™* to add an application to the system, and then assign it to a server and a user.

You can now test the application.



Note

For more information on WSM Web page settings, refer to the *Administrators Guide: Wyse WSM™*.

Testing Appsets

Testing is required to make sure that all features work in the published appset. You should complete all testing before you create the final appset. After the appset is loaded on a server, you can activate it from a client to test the features and functions of the application. *Do not* use the publishing machine for testing. Keep in mind that the prime objective of testing the application and its features is to verify that they are included in the appset package. After you complete testing and have confirmed that the published application functions completely, you can leave the appset on the server for streaming, and also install it on additional servers as needed. Refer to the *Administrators Guide: Wyse WSM™* for further details.

Testing comprises delivery testing and package testing. Delivery testing verifies that the appset is properly installed on the target system. Package testing is done to verify that all features of the target application are included in the appset. You should perform these tests on all of the target operating systems and service pack versions.

Delivery Testing

To validate an appset for a complete delivery, activate the appset and, after activation has been completed, begin the testing as described in "Application Testing." An improperly published appset may contain artifacts (for example, system drivers, broken system files) that can cause a target system to misbehave.

Use the following guidelines:

- Make sure that the application desktop and Start Menu icons appear in the appropriate locations.
- Make sure that the file associations of the existing application and the appset application work by following these guidelines:
 - Double-click a file for the appset to load.
 - Double-click a `.txt` file for Notepad to open.
- From a command line, start Notepad to verify that the Windows shell command path functions.
- Start Internet Explorer and make sure that the Windows shell does not cause Installer pop-ups to occur and that Web browsing is functional.
- Start the appset application and measure the startup time.
- Use Wordpad to make sure that all system fonts are working properly.

Application Testing

To verify that the appset contains all the necessary components of the target application, start the appset and do the following to test the application:

1. Make sure that the application started properly.
2. Click each menu item of the application to make sure that they operate properly.
3. If the application supports printing, print a document.
4. Use the application as a general user (for example, create a document and save it).
5. Exit the application to verify proper shutdown.

Correcting Common Problems

When you test an appset and encounter errors, you most likely need to republish. However, you need to know what to fix before republishing. The following sections offer tips and guidelines for correcting typical problems.

Desktop/Start Menu Icons Missing

There are two likely causes for missing program shortcuts when you subscribe to an appset.

- The move shortcuts option provided by the appset creation wizard may not have been selected. This option moves program shortcuts to the desktop or start menu upon activation.
- Exclusion of the shortcut files (.lnk) from the .aeb file. Desktop shortcuts are stored in a special folder named COMMON_DESKTOPDIRECTORY, and start menu shortcuts are stored in a special folder named COMMON_PROGRAMS. If either of these folders is present, make sure that they contain the .lnk files needed for the program.

Microsoft Icons do not Appear

Some of the newer MSI-installation based applications use special Windows Installer shortcuts. When you install these applications to publish, disable the shortcuts to a regular .lnk file. Use the command:

```
msiexec.exe /i "*.msi" DISABLEADVTSHORTCUTS=1
```

to disable them. However, if you installed and published the application without running the previous command, there will be some unwanted behavior. When you subscribe to an application that has installer shortcuts, the shortcuts appear as unknown files in the start menu (sometimes, they will not appear at all). If you click on any of them, you will more than likely get an MSI pop-up. There are two solutions to this problem:

- Run the `msiexec` command upon installation if you're just beginning to publish.
- However, if the appset has already been published, you must edit the program shortcuts manually. Create new shortcuts with the same names as the old ones and set them to target their corresponding .exe files. This can be tedious, but it is much quicker than re-installing the application using the `msiexec` command.

File Associations not Set Properly

File associations for some applications sometimes do not get set properly when you test an appset. One example would be creating a .doc file from a published version of MS Word and receiving an error message as a result. To solve this problem, go back to the publishing image, create the appropriate file type from the application being published (for example, Word), and then double-click that file to open the application. Take another snapshot, create a build, and republish.

Command Path Broken

Some applications modify the local machine's PATH environment variable. Typical applications that do this include CAD programs and IDE suites. Some system-specific entries may be included in the PATH variable upon publishing. You should take note of this when cleaning up the registry section of the build file. To view the values of the command path for the build file, do a search for path, and eventually you will locate the environment variables. Make sure that it contains no system specific entries. Client machines will have many different configurations; users may specify their own command paths. Leave only the entries that are specific to the application being published.

No Print Output or Printing Problems

In the files section of the build file, there is a folder named spool. Applications that come with printer drivers may not work properly if this folder is removed. Within the registry section is a folder named drivers that contains references to the printer drivers in the spool folder. If printing appears to cause problems in an appset, check for these resources in the build file.

System Fonts Appear Corrupted

Most applications come with their own program fonts. If subscribing/unsubscribing to an appset corrupts your system fonts and the appset uses a font that is a default system font anyway, remove all references to that particular font from the build file. The corruption is caused by the appset overwriting the registry references to the system font with that of the program fonts. By removing the font references from the build file, you allow the system fonts to be used by default.

Microsoft Installer Pop-ups Display

One common occurrence is the display of Microsoft installer pop-ups when the appset is run. This can occur if the application .msi files were removed from the build (.aeb) file. You must republish the appset in this case.

- Inspect an original non-cleaned build file or your recorded macros to see if the .msi files were excluded. Another potential solution is to remove all installation source references for the application being published. Use the Find option in the Publisher (Edit -> Find) and look for registry keys named InstallSource or SourceList.
- You can eliminate the msi pop-ups by excluding these registry keys from the appset. If the previous two methods do not work, you might have to edit the application .msi file.

You can use Orca to manually edit an .MSI file and to clear the contents of the tables named InstallExecuteSequence and InstallUISequence. This should remove the resiliency check in the local .msi file so that you can republish the appset.

Missing Files

Files are sometimes excluded during publishing, and when you run the application feature that requires the files, the application hangs or displays an error message.

- A good way to locate this missing file is to compare an original and non-cleaned-up build (.aeb) file with the cleaned-up build file (the one used to create the appset). Look through each of the excluded/removed folders and browse through the entire folder structure to confirm that it does not contain any important program files. If you know exactly what file is missing, you can search for that specific file. If you do not find it, add the file into the build file.
- A safer but more time-consuming approach is to take a new system snapshot and create a new build file with the clean snapshot. Make sure that the missing file(s) are included in the new.aeb file, and then republish.

Application Licensing Problems

Most applications employ some form of licensing. Licensing an application is commonly done with a special alphanumeric key or by obtaining a license file from the application vendor. If end users request licensing changes, for example, you will need to republish the appset. Most applications store installation information in the registry. You can search for keys named InstallProperties and license from within the .aeb file. Often you find registry entries for the User ID and serial number of the application that you can change to meet the client's request.

Miscellaneous Errors

Occasionally there are errors that can be difficult to track down. If you have the client testing machine available, you can use the Event Viewer to view a detailed description of any system/application errors that occurred during testing. You can also check the client machine for the suspected missing files to verify if they are included in the appset.

Creating and Using a Prefetch File

This section includes information on creating a prefetch file and republishing an application with it.

After you have tested the application and cleared any errors or problems, you can create a prefetch file. Prefetching improves response time when the user runs an application or common application functions for the first time. Prefetching streams the necessary appset pages in advance, thereby reducing the overhead of multiple requests to the application server.

The Publisher marks the pages to be prefetched based on a prefetch pairs file that contains a list of number pairs which identify a specific file and page offset to be prefetched. At activation time, the pairs stream down the corresponding pages. The prefetch file is created by logging pages that are being streamed down when the application is executed. This includes all the functions required by the application when the user opens it for the first time. It is recommended that you clean up the client cache before creating the prefetch file.

Cleaning Up the Client Cache

Use the following guidelines:

1. If the WSM client is currently running, stop the WSM Client (click **Start**, select **Programs**, select **WSM Client**, and then click **Stop WSM Client**).
2. Launch Windows Explorer and navigate to the folder where the WSM Client is installed (default location is: *C:\Program Files\Wyse\WSM*), then double-click on **XNetClean.bat**. If a confirmation prompt appears, click **Yes**.
3. Restart the WSM Client (click **Start**, select **Programs**, select **WSM Client**, and then click **Start WSM Client**).

Creating a Prefetch File

Use the following guidelines:

1. Create an appset for the application without using prefetching. This is typically done already as described in "Creating an Appset."
2. Publish the appset onto the Application Server as described in this chapter.
3. Subscribe to, and activate, the appset for which the prefetch pages are to be logged (refer to the *Users Guide: Wyse WSM™* for instructions on using the WSM Client to manage the applications available to users from a network server).



Note

If you have already subscribed to the application and used it at least once, you must unsubscribe, clean up the client cache (as described in "Cleaning Up the Client Cache"), and restart your system. Then return to this step to subscribe to, and activate, the appset for which the prefetch pages are to be logged.

4. Start logging the streamed down pages by running the following command in a DOS or Command Prompt window from the WSM Client installation folder (default location is: *C:\Program Files\Wyse\WSM*):
`eSUser.exe "beginlog=<prefetch_file_patch>"`
 where <prefetch_file_path> is the full path and file name of the prefetch file to be created (for example, *C:\prefetch.log*). Note that if you do not specify a path, the Windows system folder is used by default, and if a relative path is specified, it is taken relative to the Windows system folder as well. Also note that the double quotes are only necessary if the path contains spaces.
5. Execute the application.



Note

The prefetch data should include the code needed to start and terminate the application and for frequently used commands. For example, the opening and saving of documents are frequently used commands and should be prefetched if possible. Other items to include for prefetch are those associated with frequently accessed folders and associated file meta data.

6. (If you only want to collect data for application startup, you can skip this step) Execute the functions of the components to be included with the prefetch.



Note

To prefetch the open and save commands, first create a new document save it somewhere other than the O: drive, then re-open it. Do not save any

documents on the O: drive, or open any application samples from the O: drive, as this will create unwanted artifacts when you re-publish the application.

7. Exit the application.
8. To stop the prefetch logging, run the following command from the WSM Client installation folder:
`eSUser.exe endlog`

Republishing an Appset with a Prefetch File

If you have created a prefetch file (as described in "Creating a Prefetch File") you can republish an application with it.

Use the following guidelines:

1. Ensure the prefetch file is accessible on the publishing machine.
2. Launch the Publisher from the *Start* menu.
3. On the *File* menu, select **Open Build**, select the edited build file saved during the original generation of the appset, and then click **Open**.
4. On the *File* menu, click **Create Application Set** to open the **Create an Application Set** dialog box.



Note

The publishing process may be stopped and the following error message will display if you have made modifications to the build and the prefetch file is invalid after the modifications:

"Line xxxx in <prefetch file> Page number x is not a part of file #xx in Xnet set....."Error with creation of <.aes filename> internal error: cannot parse and validate prefetch file".

In this situation, you must republish without the prefetch, recreate the prefetch file, create a new appset with the new prefetch, and then republish.

5. Select **Load an existing application set configuration (.aec)**, click **Browse**, select the configuration file saved during the initial publishing of the application, and then click **Open**.
6. (Optional) If you intend to keep the old appset on the application server alongside the new appset, click **Next** until you reach the *Main Options* page, click **Generate GUID** next to the *Application set ID* field. Otherwise, skip this step.



Note

Unless you intend to overwrite the old appset, you must generate a new appset ID in order to prevent conflicts between the two appsets. If you want to use the same ID for the new appset, then remove the old appset from the application server before publishing the new one.

7. Click **Next** until you reach the *Advanced Options* page.
8. Click **Browse** next to the *Prefetch file path* field, select the prefetch file previously generated, and then click **Open**.
9. Create the appset as described in "Creating an Appset."
10. Publish the new appset to the WSM server.

Modifying and Updating Appsets

There are many reasons why you may want to make changes to an existing appset, including:

- Modifying the list of supported operating systems
- Adding, removing, or modifying AppEvent DLLs
- Adding, removing, or modifying files and registry entries
- Modifying appset configuration parameters

Some of these changes require that an application be republished, as described in "Republishing an Application." Because republishing can be time-consuming, WSM Publisher offers the ability to make certain changes without the need to republish the application (the first three items listed do not require republishing).

Modifying the List of Supported Operating Systems

An appset may be initially marked to run on the OS of the publishing system. Later, testing may reveal that the appset may also run properly on other operating systems, or other OS/SP combinations. Conversely, testing may reveal that the appset does not run properly on an OS for which it was marked. In such cases, you need to modify the target operating systems that are marked for the appset.

To modify the supported operating systems:

1. From the Tools menu, select **Update OS Support**.

The Supported Operating Systems dialog appears.

2. Click **Browse**, navigate to the location of the appset that you want to modify, and double-click it.

The content type of the selected appset will be shown as Unicode, Non-unicode, or Unknown.

3. Select the supported OS/SP combinations as described in "Step 2: Selecting Operating Systems."
4. Click **Update** to initiate the update process.

Adding, Removing, or Modifying AppEvent DLLs

After publishing an application, appset testing may reveal some problems that cannot be resolved by simply changing the build file. In this case, you may have to use an AppEvent DLL to perform special actions. You may also wish to change or remove an existing AppEvent DLL.

To add, change, or remove an AppEvent DLL:

1. From the Tools menu, select **Update Custom DLL**.

The Update Custom DLL in Application Set File dialog appears.

2. Do one of the following:

- If you want to remove an existing AppEvent DLL, select the **Remove Custom DLL Only** option.
- If you want to add new AppEvent DLL or change an existing one, select the **Custom DLL File** option and then specify the path of the new AppEvent DLL file in the field.

3. Specify the path of the appset file that you want to update in the Target Application Set File field.
4. Make sure that the **New Application Set ID** option is selected. If you want to change the ID that is automatically generated for you, click **Generate**.

**Caution**

You should always generate a new ID for an appset, even if you intend to overwrite the old appset on the application server. Although you could use the same appset ID if you first remove the old appset from the application server, doing so would invalidate all subscriptions associated with the old appset! So, if your intent is to upgrade existing users to the new appset, you should use the upgrade process documented in the *Administrators Guide: Wyse WSM™* instead. One situation where you may actually want to use the same appset ID is for testing purposes only, but never for a released appset.

5. Click **Start** to initiate the update process.

Republishing an Application

Republishing an application is the process of creating a new appset from part or all of the data used to create the original appset for the application. As indicated earlier, not all changes to an appset require that an application be republished. However, it may be necessary to do so if the target application does not function as intended (because one or more items were removed from the build file but should have remained or an item was left in but should have been removed). You may also need to republish an application for licensing changes or to change the functionality of the application. There are two ways to republish, depending on the change required:

1. If you discover that you did not remove a particular file or registry entry from the build, you can open the existing appset with the Import build from Application Set feature (*File* menu). This creates a temporary build file from the contents of the appset. You can then modify the file or registry and republish from the new build file. Because the appset already contains all the files necessary for the application, you do not need to use the machine image of the publishing machine that was used to create the appset.
2. Other modifications require the machine image of the publishing machine. You can use imaging utilities such as Norton Ghost to recall machine images. After restoring the machine image, you can modify the appset. For example, if a user wants an application to open to a certain view at startup, you can execute the program from the publishing machine, set the default properties from within the application, take a new snapshot, and then construct a new build file. You would then execute the saved macros on the new build file, republish, and test the appset based on the new build file.

Tips for republishing:

- Use your saved macros to republish an application. They will save time and help avoid errors that can occur from manual clean-up.
- When you republish with an appset, open the appset within the fastest machine available. Faster hardware is a big-time saver.
- If no macros were saved from the initial publishing process, you can open an old build file for the application and use it as a guide for cleaning the build file.
- Always change the Application ID (not the GUID) of the appset you are republishing. This eliminates any conflicts in the application server if you want to test different versions of the appset at the same time.

This page intentionally blank.

8

Publisher Macros

This chapter describes the macros provided with WSM Publisher.

Although macros can save considerable time in cleaning-up build files, knowing exactly what to take out and what to leave in an appset comes with experience. If you discover that you removed something that should have remained, you can simply republish the appset. If you used a macro or created one while cleaning a file, you can modify the macro to reflect the required inclusions or exclusions and then run the macro on the build file for a new appset.

Additional benefits of macros include:

- Less chance of error (compared to manual modification) when an application is republished.
- The ability to include comments for a macro about what can be and should be removed so that others can republish an application using the macro.

It is recommended that you create macros when publishing a new application. If there is a macro for an older version of the application, you can use it as a guide to create a new macro, as newer versions of applications usually require similar exclusions or removals.

Macro Example

The following example macro is an overview of what can be found within a macro file:

```
Sub SampleMacro()  
CurrentDoc.DeleteFile "COOKIES"  
CurrentDoc.SetAccessToken "INSTALLDIR", True, True, True  
CurrentDoc.DeleteReg "HKEY_LOCAL_MACHINE\HARDWARE"  
End Sub
```

Publisher macros are written in Microsoft VBScript. For more information on VBScript, refer to the VBScript documentation in the Microsoft Developer Network (MSDN) Library at this address: <http://msdn.microsoft.com/library>.

The CurrentDoc scripting object is used by WSM Publisher to manipulate build files.

This particular macro conducts both a file and registry entry deletion along with the enabling of access tokens. The line `CurrentDoc.DeleteFile "COOKIES"` removes the special folder named "COOKIES" from the build file. The line `CurrentDoc.SetAccessToken "INSTALLDIR", True, True, True` sets access tokens for all .exe files found within the INSTALLDIR folder in the build file. Finally, the line `CurrentDoc.DeleteReg "HKEY_LOCAL_MACHINE\HARDWARE"` removes the aforementioned registry key from the build file. The CurrentDoc functions have been encapsulated in a function called `SampleMacro()`. The last line simply executes all the commands found within `SampleMacro()`.

This is a macro at its most basic form. As the publisher, you can use more advanced features of VBScript to create more complex and robust macros. Documentation for VBScript can be found in the Microsoft Developer Network Library (MSDN).

Creating and Editing Publisher Macros

The easiest way to create a new macro is by recording it using WSM Publisher, as described in "Recording Macros." However, you can also use any text or VBScript editor to create or modify a macro. WSM Publisher also provides a built-in VBScript editor that allows you to modify existing macros. The Macro Editor features syntax coloring, objects event listing, pop-up object properties and methods list, and function parameters tool-tips.

✓ Note

While editing macros through VB Script Editor and the object event is changed for CustomAppEvent Macro and Registry_Sourcelist Macro, a *Microsoft Send Error Message* will be displayed. In the same case, when you click **Don't Send**, WSM Publisher will close.

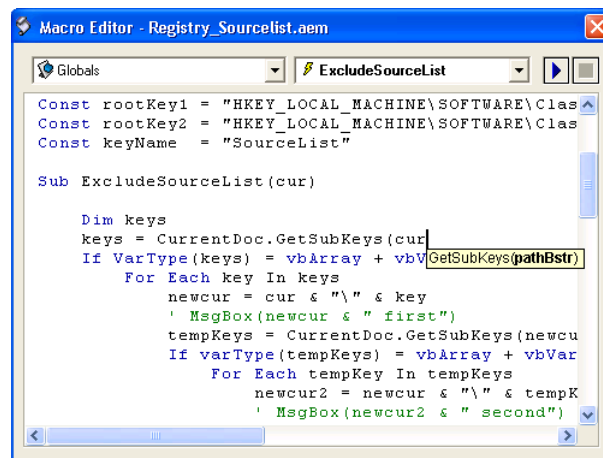
In addition, when the Object Event is changed to *addEventCommand / doMacro* for CustomAppEvent Macro or when the Object Event is changed to *ExcludeInstallProperties for Registry_Sourcelist Macro*, a *Microsoft Send Error Message* will be displayed.

Editing a Macro Using the Built-in VBScript Editor

To edit a macro using the built-in VBScript editor:

1. From the Macro menu, select **Play Macro**.
2. Add the macro to the list, as described in "Using Macros for Automated Clean-up."
3. Select the macro, and then click **Edit Macro** (it appears in the built-in Macro Editor).

Figure 33 Macro Editor



4. Make your changes, click the Close button [X], and then click **Yes** in the pop-up window to save your changes.

Macro References

This section includes Macro reference information on CurrentDoc Object and CurrentDoc Methods.

CurrentDoc Object

The Publisher exposes only one scripting object (CurrentDoc) to manipulate build files.

The CurrentDoc scripting object is used to manipulate the currently loaded build file and perform file and registry-entry operations that are available in the Publisher UI. It has no properties and no events, and supports the methods described in the next section.

CurrentDoc Methods

The CurrentDoc object supports the following methods that manipulate the currently loaded build file (the CurrentDoc scripting object supports the following methods that manipulate the currently loaded AEB):

- AddDirTree
- AddFile
- AddReg
- AddRegTree
- DeleteFile
- DeleteReg
- ExcludeFile
- ExcludeReg
- GetFiles
- GetRegType
- GetRegValue
- GetRegValueNames
- GetSubDirs
- GetSubKeys
- SetAccessToken
- SetAimDisposition
- SetRegType

AddDirTree Method

Description

Recursively adds a complete folder (directory) tree from disk to the current build file.

Syntax

AddDirTree *path*

The **AddDirTree** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full, non-parameterized path to the folder to be added to the AEB.

Return Values

The **AddDirTree** method has no return values.

Remarks

The following example uses **AddDirTree** to add `c:\winnt\system32` to an AEB on a Windows XP machine:

```
CurrentDoc.AddDirTree "c:\winnt\system32"
```

After running the command, the SYSTEM32 folder and all its contents will appear in the AEB.

AddFile Method

Description

Adds a file or folder from disk to the current AEB.

Syntax

AddFile *path*

The **AddFile** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full, non-parameterized path to the file or folder to be added to the AEB.

Return Values

The **AddFile** method has no return values.

Remarks

The following example uses **AddFile** to add `c:\winnt\system32\shell32.dll` to an AEB on a Windows XP machine:

```
CurrentDoc.AddFile "c:\winnt\system32\shell32.dll"
```

After running the command, the SYSTEM\shell32.dll will appear in the AEB.

AddReg Method

Description

Adds a registry key and/or value to the current AEB.

Syntax

AddReg *keyPath*, *hasValue*, *valueName*, *dataType*, *valueData*

The **AddReg** method takes the following parameters:

Parameter	Description
keyPath	Required <i>String</i> . Specifies the full path to the registry key to be added to the AEB. If a value is being added, this parameter specifies the value's parent. If the parent key does not exist, it will be created.
hasValue	Required <i>Boolean</i> . Specifies whether or not a value is being added in addition to the key specified in <i>keyPath</i> . If this value is false, the <i>valueName</i> , <i>dataType</i> and <i>valueData</i> parameters are ignored (although they must be given values).
valueName	Required <i>String</i> . Specifies the name of the registry value to add to under the key specified by <i>keyPath</i> . Use an empty string ("") to specify the default value.
dataType	Required <i>Integer</i> . Specifies the data type of the value to be added. Can take one of the following values: Value Name 0 REG_NONE 1 REG_SZ 2 REG_EXPAND_SZ 3 REG_BINARY 4 REG_DWORD 5 REG_DWORD_BIG_ENDIAN 6 REG_LINK 7 REG_MULTI_SZ 8 REG_RESOURCE_LIST
valueData	Required <i>String</i> . Specifies the actual data to assign to the value to be added. Must be in the following format: REG_SZ and REG_EXPAND_SZ: A string (for example, "this is a REG_SZ value"). REG_DWORD: A decimal integer in quotes (for example, "67"). All others: A string of space separated hex bytes (for example, the 32-bit REG_BINARY value 0xB0BDB0BD would be added as "B0 BD B0 BD").

Return Values

The **AddReg** method has no return values.

Remarks

The following example uses **AddReg** to add HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\AppPaths\Acrobat.exe and its default value to an AEB on a Windows XP machine:

```
CurrentDoc.AddReg
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App
Paths\Acrobat.exe",
true, "", 1, "z:\Program Files\Adobe\Acrobat 4.0\Acrobat\Acro-
bat.exe"
```

AddRegTree Method**Description**

Adds the entire registry tree rooted at the specified key to the current AEB.

Syntax

AddRegTree *keyPath*

The **AddRegTree** method takes the following parameters:

Parameter	Description
keyPath	Required <i>String</i> . Specifies the full path to the root key of the registry tree to be added to the AEB.

Return Values

The **AddRegTree** method has no return values.

Remarks

The following example uses **AddRegTree** to add the entire registry tree rooted at HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion to an AEB on a Windows XP machine:

```
CurrentDoc.AddRegTree "HKEY_LOCAL_MACHINE\SOFTWARE\Micro-
soft\Windows\CurrentVersion"
```

DeleteFile Method**Description**

Completely removes a file or folder tree from the current AEB.

Syntax

DeleteFile *path*

The **DeleteFile** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full parameterized path to the file or folder to be removed from the AEB.

Return Values

The **DeleteFile** method has no return values.

Remarks

The following example uses DeleteFile to remove SYSTEM\shell32.dll from an AEB on a Windows XP machine:

```
CurrentDoc.DeleteFile "SYSTEM\shell32.dll"
```

After running the command, the shell32.dll file will be removed from the SYSTEM folder.

DeleteReg Method

Description

Completely removes a value or registry key from the current AEB.

Syntax

DeleteReg *path*

The **DeleteReg** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full path to the value or key to be removed from the AEB.

Return Values

The **DeleteReg** method has no return values.

Remarks

The following example use DeleteReg to remove HKEY_LOCAL_MACHINE\Software from an AEB on a Windows XP machine:

```
CurrentDoc.DeleteReg "HKEY_LOCAL_MACHINE\Software"
```

After running the command, the entire Software tree will be remove from the registry changes area of the current AEB.

ExcludeFile Method

Description

Marks a file or folder for exclusion from, or inclusion in the appset.

Syntax

ExcludeFile *path*, *excludeFlag*, *recursiveFlag*

The ExcludeFile method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full, parameterized path to a file or folder to be excluded from, or included in the appset.
excludeFlag	Required <i>Boolean</i> . Specifies whether or not the item should be excluded in the appset. If set to true, the item is not included in the appset. If set to false, then the item is included in the appset.
recursiveFlag	Required <i>Boolean</i> . If the path represents a folder, this specifies that the ExcludeFile operation should be performed on all the files and folders in the subtree under the folder. Note that if you are excluding a folder (that is, <i>excludeFlag</i> is set to true), the operation is automatically recursive. That is, you cannot exclude a parent folder without also excluding its children. On the other hand, you may wish to include back an excluded folder without necessarily including its children. In this case, set the recursive parameter to false. Ignored when path specifies a file.

Return Values

The **ExcludeFile** method has no return values.

Remarks

The following example use **ExcludeFile** to exclude the SYSTEM\drivers folder and all its contents from packaging into the appset:

```
CurrentDoc.ExcludeFile "SYSTEM\drivers", true, true
```

After running the command, the entire drivers folder will be marked for exclusion.

Use **DeleteFile** to permanently delete file entries from the AEB.

ExcludeReg Method

Description

Marks a registry key or value for exclusion from, or inclusion in the appset.

Syntax

ExcludeReg *path*, *excludeFlag*, *recursiveFlag*

The **ExcludeReg** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full path to a key or value to be excluded from, or included in the appset.
excludeFlag	Required <i>Boolean</i> . Specifies whether or not the item should be excluded in the appset. If set to true, the item is not included in the appset. If set to false, then the item is included in the appset.
recursiveFlag	Required <i>Boolean</i> . If the path represents a key, this specifies that the ExcludeReg operation should be performed on all the keys and values in the subtree under the key. Note that if you are excluding a key (that is, <i>excludeFlag</i> is set to true), the operation is automatically recursive. That is, you cannot exclude a parent key without also excluding its children. On the other hand, you may wish to include back an excluded folder without necessarily including its children. In this case, set the recursive parameter to false. Ignored when path specifies a value.

Return Values

The **ExcludeReg** method has no return values.

Remarks

The following example use ExcludeReg to exclude the HKEY_LOCAL_MACHINE\System\ControlSet001 key and all its contents from packaging into the appset:

```
CurrentDoc.ExcludeReg
"HKEY_LOCAL_MACHINE\SystemControlSet001", true, true
```

After running the command, the entire ControlSet001 key will be marked for exclusion.

Use **DeleteReg** to permanently delete registry entries from the AEB.

GetFiles Method

Description

Retrieves a list of strings containing the names of all the files in a folder.

Syntax

```
outArray = GetFiles path
```

The **GetFiles** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full, parameterized path to the folder from which to get the file names.

Return Values

The **GetFiles** method returns an array of strings containing the names of the files.

Remarks

The following example uses **GetFiles** to display the names of the first four files under the INSTALLDIR folder:

```
dim files

files = CurrentDoc.GetFiles "INSTALLDIR"

MsgBox(files(0))

MsgBox(files(1))

MsgBox(files(2))

MsgBox(files(3))
```

GetRegType Method

Description

Retrieves an integer representation of the data type of a specified registry value.

Syntax

```
dataType = GetRegType path
```

The **GetRegType** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full path to the registry value from which to get the data type. For the default value of a key, specify "KEY\<default>".

Return Values

The **GetRegType** returns an integer containing the data type.

Remarks

The following example uses **GetRegType** to display the data type of the MyData value under the HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp registry key

```

dim dataType

dataType = CurrentDoc.GetRegType
"HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp\MyData"

MsgBox("The dataType is " & Str(value))
    
```

GetRegValue Method

Description

Retrieves the string representation of a specified registry value.

Syntax

```
outString = GetRegValue path
```

The **GetRegValue** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full path to the registry value to retrieve. For the default value of a key, specify "KEY\<default>"

Return Values

The **GetRegValue** returns a string containing the registry value. The format is the same as that described in AddReg. To interpret the value, you may need to call GetRegType.

Remarks

The following example uses **GetRegValue** to display the data in the MyData value under the HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp registry key:

```

dim value

value = CurrentDoc.GetRegValue "HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp\MyData"

MsgBox("The value is " & value)
    
```

GetRegValueNames Method

Description

Get a list of strings containing the names of each of the values under a registry key.

Syntax

outArray = **GetRegValueNames** *path*

The **GetRegValueNames** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full path to the registry key from which to get the value names.

Return Values

The **GetRegValueNames** returns an array of strings containing the names of the values.

Remarks

The following example uses **GetRegValueNames** to display the names of the first four values under the HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp registry key:

```

dim files

values = CurrentDoc.GetRegValueNames
"HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp"

MsgBox(values(0))

MsgBox(values(1))

MsgBox(values(2))

MsgBox(values(3))

```

GetSubDirs Method

Description

Retrieves a list of strings containing the names of all the subfolders under a folder.

Syntax

outArray = **GetSubDirs** *path*

The **GetSubDirs** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full, parameterized path to the folder from which to get the subfolder names.

Return Values

The **GetSubDirs** returns an array of strings containing the names of the subfolders.

Remarks

The following example use **GetSubDirs** to display the names of the first four subfolders under the INSTALLDIR folder:

```
dim dirs

dirs = CurrentDoc.GetSubDirs "INSTALLDIR"

MsgBox(dirs(0))

MsgBox(dirs(1))

MsgBox(dirs(2))

MsgBox(dirs(3))
```

GetSubKeys Method**Description**

Retrieves a list of strings containing the names of all the subkeys under a key.

Syntax

outArray = **GetSubKeys** *path*

The **GetSubKeys** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full path to the key from which to get the subkey names.

Return Values

The **GetSubKeys** returns an array of strings containing the names of the subkeys.

Remarks

The following example use **GetSubKeys** to display the names of the first four subkeys under the HKEY_LOCAL_MACHINE registry key:

```
dim keys

keys = CurrentDoc.GetSubKeys "HKEY_LOCAL_MACHINE"

MsgBox(keys(0))

MsgBox(keys(1))

MsgBox(keys(2))

MsgBox(keys(3))
```

SetAccessToken Method

Description

Sets the Access Token flag for a file or folder tree.

Syntax

SetAccessToken *path, accessTokenFlag, recursiveFlag, exesOnlyFlag*

The **SetAccessToken** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full, parameterized path to a file or folder for which to set the Access Token flag.
accessTokenFlag	Required <i>Boolean</i> . Specifies the new value for the Access Token flag. If it is true, the flag will be set. If it is false, the flag will be cleared.
recursiveFlag	Required <i>Boolean</i> . If the path represents a folder, this specifies if the SetAccessToken operation should be performed on all the files in the subtree under the folder. Ignored when path specifies a file
exesOnlyFlag	Required <i>Boolean</i> . If the path represents a folder and <i>recursiveFlag</i> is set to true, this specifies that the SetAccessToken operation should be performed only on files with a .exe extension.

Return Values

The **SetAccessToken** method has no return values.

Remarks

The following example uses **SetAccessToken** to set the Access Token flag for all the EXEs in the PROGRAM_FILES folder to true:

```
CurrentDoc.SetAccessToken "PROGRAM_FILES", true, true, true
```

SetAimDisposition Method

Description

Sets the Client Disposition value for a file or folder tree.



Note

AIM refers to the WSM Client's Application Installation Manager.

Syntax

SetAimDisposition *path, dispositionType, recursiveFlag*

The **SetAimDisposition** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full, parameterized path to a file or folder for which want to change the Client Disposition.
dispositionType	Required <i>Integer</i> . Specifies the new Client Disposition type: Value Name 0 AIM_SPOOF 1 AIM_COPY 2 AIM_INSTALL
recursiveFlag	Required <i>Boolean</i> . If the path represents a folder, this specifies if the SetAimDisposition operation should be performed on all the files in the subtree under the folder. Ignored when path specifies a file.

Return Values

The **SetAimDisposition** method has no return values.

Remarks

The following example uses **SetAimDisposition** to change the Client Disposition flag on SYSTEM\drivers folder and all its contents to AIM_COPY:

```
CurrentDoc.SetAimDisposition "SYSTEM\drivers", 1, true
```

After running the command, the entire drivers folder will be marked for copy to the client by the AIM (Application Installation Manager).

SetRegType Method

Description

Changes the data type of a registry value.

Syntax

```
SetRegType path, newType
```

The **SetRegType** method takes the following parameters:

Parameter	Description
path	Required <i>String</i> . Specifies the full path to the value for which to change the data type. For default values of a key, specify "KEY<default>".
newType	Required <i>Integer</i> . Specifies the new type, using the constants defined in AddReg.

Return Values

The **SetRegType** has no return value.

Remarks

When the type of a registry value is modified, the value itself does not change. Instead, it is re-interpreted according to the new type.

The following example uses **SetRegType** to set the data type of the MyData value under the HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp registry key to REG_BINARY:

```
CurrentDoc.SetRegType "HKEY_LOCAL_MACHINE\Software\MyCompany\MyApp\MyData", 3
```

This page intentionally blank.



9

Case Studies

This chapter provides two case studies to demonstrate the publishing and cleanup procedures and processes applied to specific applications.

The first case study uses Macromedia Dreamweaver to represent simpler and more typical publishing. The second case study, Publishing Microsoft Word 2003 on Windows XP SP1 demonstrates the installation of Microsoft products (which generally involve a more complex installation process).

You can print out the case studies for convenience. To do this, right-click the book icon for the case study and select Print. To print both case studies, right-click the Case Studies book icon and select Print.

Publishing Macromedia Dreamweaver MX

Dreamweaver MX is a relatively small application without a high demand for operating system resources. This case study describes the main requirements for publishing Dreamweaver MX, including pre-configuring the system for publishing. You can publish Dreamweaver to function properly in earlier Windows operating systems with the WSM Publisher appset creation wizard, however, this case study is based on Windows XP.

General Requirements for Publishing

These requirements apply to the publication of any application.

- Set up a dedicated machine for publishing. A dedicated machine will have a clean OS image and WSM Publisher. Make sure that the OS is patched with all the latest updates from the Microsoft update site.
- The dedicated machine should have a physical or logical partitioned drive (O:). All files can be stored and published on this drive. You can create a folder to store all the files generated from the Publisher into one location.

Preparing for Publishing

Follow these guidelines to prepare for publishing:

- Using the Publisher, create the installation folder and configuration file (.aec). For example: O:\Macromedia Dreamweaver MX\INSTALLDIR\

Installation folders are created for Dreamweaver MX at a specific location so that the Publisher can reference them.

- Before closing the Create Installation Folder window, select **Save Config...** and save the configuration file on the O drive for later use in publishing.

Taking Snapshots

Two snapshots are required. The first captures the system before the installation; the second captures the changes after the application has been installed and run once. If the application installation file is on the CD, be sure it is in the drive before you start the installation. If the installation files are not on a CD, you can put the files in a separate folder on the O drive.

- To reduce the artifacts in the snapshot, make sure the CD is in the drive before you take the first snapshot.
- To avoid artifacts during installation, do not open and close other applications or windows if possible.
- As good practice, include the sequence in the snapshot file label; for example, snap1.ss to indicate the first snapshot and snap2.ss to indicate the second snapshot after installation.
- Before you start the installation, take the first snapshot and label the file accordingly.

Installing Dreamweaver MX

1. After the first snapshot, install Dreamweaver MX. Go through with the installation wizard and select the features to be installed. Keep in mind that some features in Dreamweaver MX may not function if not installed, so it is always better to have a complete and full installation.
2. When you are prompted for a location for the installation, specify the installation folder created earlier. (that is, "O:\Macromedia Dreamweaver MX\INSTALLDIR").
3. Continue and finish the installation wizard to install the application.
4. Launch Dreamweaver for the first time.
5. Take a second snapshot to capture the system changes that were made by installing the application.
6. To determine the changes made by the installation, compare the first and second snapshot.

Cleaning Files in the .aeb File

After you compare the snapshots, you can go through the file and registry to determine what files to exclude from the build or to remove entirely.



Note

You could use macros to do much of the clean-up. However, the steps in the clean-up process are delineated here for instructional purposes.

1. Start recording a File Macro. The actions you should record will be performed in the Files section of the .aeb file.
2. Proceed through the file and registry exclusions:

Files:

- The APPDATA folder corresponds to the C:\Documents and Settings\User\Application Data folder in the publishing machine. The Identity Folders is the identity setting for the user on the publishing machine but is found empty and can be removed. The Macromedia folder has files installed for the user after installation of Dreamweaver; this folder can be left alone.
- The COMMON_PROGRAMS folder contains all the program shortcuts for Macromedia Dreamweaver MX and can be left alone unless there are links that the Publisher wants to exclude.
- The COMMON_STARTMENU folder contains links to the program for all other users, but the links found in COMMON_PROGRAMS will be populated by default. This folder is empty and can be removed.
- The COOKIES folder should always be removed. It contains files generated by Internet Explorer and is unnecessary when publishing any appset.
- The HISTORY folder contains more files used by Internet Explorer. Remove this folder completely.
- The INSTALLDIR folder corresponds to the installation folder we created with the Publisher. It contains the main program files that will be used by Dreamweaver MX. No exclusions or removals should be made on its contents. The only task we need to do here is to set access tokens for .exe files (.dll files can also be set with access tokens).
- The INTERNET_CACHE folder corresponds to the Temporary Internet Files folder in the publishing machine. Remove this folder completely.
- O: corresponds to the Publishing drive. This folder is never needed and should be removed. All the vital files in this drive are already included in the INSTALLDIR folder.
- The PROFILE folder contains files and folders that store personal preferences and application-configuration information. This folder is a shared system folder and should be removed so that we do not supersede a client machine's corresponding folder.
- The PROGRAM_FILES folder contains folders that should be left alone and ones that can be safely removed. The Common Files folder is empty and can be removed. The Wyse folder contains files that are modified from the Publisher and do not need to be included in the appset, so this folder can also be removed. The InstallShield Installation Information folder is a temporary folder created to install Dreamweaver and can be removed. Uninstall Information can also be removed since it contains empty folders.
- The PROGRAM_FILES_COMMON folder corresponds to the "Program Files\Common Files" folder in the Publishing machine. It contains Macromedia Dreamweaver MX files that were installed. Setting access tokens is all that should be done here.
- The SYSTEM folder contains files and folders that should be removed or left alone depending on the application being published. This folder corresponds to the System32 folder in the Publishing machine.

Config contains files that store system and application settings. These are machine-specific and should be removed. Leaving them in may render a client machine unable to function properly.

Ias is the Windows Internet Authentication Service folder. It does not contain any files that are essential for Dreamweaver MX to function properly. This folder should be removed.

Inetsrv contains a subfolder called MetaBack. Windows stores backups of the Windows IIS Metabase in this folder. The metabase is used by Internet Explorer for fast retrieval of IIS configuration information. It is not needed for Dreamweaver MX and should be removed.

Wbem contains log files used by Windows Management Instrumentation. This is usually not needed for publishing and can be removed.

SYSTEMDRIVE corresponds to the root drive of the publishing machine (C:). It does not contain anything that would affect Dreamweaver MX functionality. It can be removed from the build.

The WINDOWS folder contains folders that were mainly modified and none were created during installation that would not affect Macromedia Dreamweaver MX functionality. Everything in this folder can be removed.

The WINDOWS folder itself contains several files that can be removed. All the log files can be removed safely since they are regenerated by Windows. Leave all the other files within this folder as is. Set the access tokens for the entire folder. This completes the cleanup of the Files section of the build file. Stop recording the macro and save it.

Cleaning Registry Settings in the .aeb File

The registry entries being removed or excluded have been categorized under one of the major root keys that are picked up when publishing.

Start recording Macro again for the removal or Exclusion during the registry.

HKEY_CURRENT_USER

- Software\Wyse contains registry keys related to WSM Publisher. No traces of the Publisher should be included in any appset being published. This key should always be removed.
- Software\Macromedia contains registry entries that are relevant to Dreamweaver MX settings and functionality. Nothing in this folder should be removed or excluded.
- Software\Microsoft contains several registry entries that may or may not be necessary for publishing.

Windows\CurrentVersion\Explorer contains registry entries related to recent Windows Explorer activity. There are keys that may be needed for the application being published; however, the following entries listed can be removed from the .aeb file:

ComDlg32 stores information on opened and saved files that were recorded by the OS during the publishing process. This should be removed.

FileExt contains any new file extensions that were created during the publishing process. Remove any other file extension not related to the application being published.

MenuOrder maintains start menu shortcut order on the publishing machine. Removing this will have no effect on the application being published.

RunMRU contains a list of recent commands executed through the Run dialog box on the publishing machine. This should be removed.

Streams stores information about the size and location of closed Windows used during the publishing process. This also must be removed.

UserAssist contains user assistant history files. This can be safely removed for any application being published.

Windows\ShellNoRoam contains information regarding recently run programs and their windows sizes and positions. Remove this to prevent overriding a local client machine's own settings.

HKEY_LOCAL_MACHINE

- Software\Classes contains information about an application that is needed to support COM functionality. Dreamweaver MX will reference this for its key extensions to open function.
- Software\Macromedia contains information on Dreamweaver MX installation and resource location. These keys should not be removed.
- Software\Microsoft contains several keys that should be removed (described below).

Cryptography stores local digital signature settings specific to the publishing machine. This can be removed.

Windows contains registry keys that should be removed (covered in greater detail below):

- App Paths contains specific paths that are needed for Dreamweaver MX
- SharedDLLs contains dll information used in the InstallShield that does not be included in the appset. This key can be removed.
- Uninstall stores information that Dreamweaver MX will need to uninstall itself from the publishing machine. The information here is system specific. This entire key should be excluded from the appset.

HKEY_USERS

This registry rootkey stores user-specific settings for the publishing machine. Information stored here is not necessary for Dreamweaver MX to function properly. The entire HKEY_USERS key can be removed completely from the .aeb file.

At this point stop recording the registry macro and save it for later use. You may also save the current build (.aeb) of all changes that are made to the appset.

Conclusion

Now that clean-up is complete, save the build file. This is the file that will be used to create the appset.

- After saving, select Create application set from the File menu. You can then create a new config file or load a config file.

In this case study, the configuration file was saved when the installation folder was created, so that can be used. If for some reason you needed to generate a new config file, you would have to use the GUID associated with the application.

- From this point, you can specify additional information about the appset to be created. You can optimize the appset with compression, encryption, and the inclusion of a specific prefetch file when recreating the appset for republishing. The Publisher creates the appset after you enter the required information in the wizard.

Publishing Microsoft Word 2003

This case study is presented for installing Microsoft Word 2003 on Windows XP SP1. Microsoft products can be a little more difficult to publish than other applications in part because Microsoft applications tend to integrate themselves deeply into the Windows OS. This can make it more difficult to determine which files, folders, and registry entries should be removed to clean up the .aeb file. This case study is presented to clarify the steps and tasks to install Word 2003 on Windows XP SP1, and to increase your overall understanding of publishing that can be applied to similar situations.

General Requirements for Publishing

These requirements apply to the publication of any application.

- If you have not already set up a dedicated machine for publishing, you should do so. A dedicated machine has a clean OS image and WSM Publisher. Make sure that the OS is patched with the latest updates from Microsoft.
- The dedicated machine should have a physical or logical partitioned drive (O:). All files can be stored and published on this drive. You can create a folder to store all the files generated from the Publisher into one location.

Preparation

The following tasks apply to publishing MS Word 2003 on Windows XP:

- Using the Publisher, create the installation folder and configuration file (.aec).
- If you are installing Word from a CD, make sure that it is in the CD-ROM drive before you take the first snapshot. This reduces the artifacts that can be captured during publishing.
- When everything is in order, create a system snapshot.
- Before closing the Create Installation Folder window, select **Save Config...** and save the configuration file on the O:\ drive for later use in publishing.

Installing Word

To ensure a reliable installation, install the Microsoft application with all of its features. This eliminates the need for the Installer functions.



Note

You cannot install Microsoft products to publish by running their setup programs. They use Windows Installer Shortcuts that are different from regular.Ink files in that they support Windows Installer's install-on-demand functions.

1. Run the `msiexec` command on Word's .msi file to install it. You must also provide a parameter to the command to disable the Installer shortcuts. You must execute the following command through the Run dialog box (**Start Menu > Run**):

```
msiexec /i "*.msi" DISABLEADVTSHORTCUTS=1
```

*.msi" should be the path to Word's .msi file. This is easy to determine by browsing the CD. This command calls the .msi file which runs Word's setup program.

2. Select a custom install and specify the GUID install folder you created for the installation folder. Also, make sure that all features of Word are set to be run locally from my computer.
3. After installation is complete, check for Word program updates from the Microsoft Office Updates page. Download and install all available updates and patches. With that done, run MS Word once to make sure that any other first-run settings are in place.
4. Take the second snapshot and create a build (.aeb) file. Save this clean build.

Cleaning Files in the .aeb File

Use the following guidelines:



Note

You could use macros to do much of the clean-up. However, the steps in the clean-up process are delineated here for instructional purposes.

- Start recording a File Macro. The actions to be recorded are in the Files section of the .aeb file. Now we can start going through the file and registry exclusions.
- The APPDATA folder corresponds to the C:\Documents and Settings\User\Application Data folder in the publishing machine. It usually contains program settings for Word. Even though most of the folders are empty, they will be used by Word.
- The only folder to remove is APPDATA\Microsoft\Office\Recent; it contains recently accessed files within Word.
- Under the APPDATA folder is an empty folder named Microsoft Web Folders. If you check the change type attribute of this folder in the Publisher, it says Created. Therefore, even though it is empty, you will leave it in. There should be no more exclusions within the APPDATA folder.
- The COMMON_PROGRAMS folder contains all the program shortcuts for Microsoft Word. The only folder that you should exclude here is the Startup folder, which is empty and is not used by Word.
- The COMMON_STARTUP folder contains links to programs that should be run at startup. In this case, it contains a link to Microsoft Office. Leave this folder alone.
- The COOKIES folder should always be removed. It contains files generated by Internet Explorer and is unnecessary for publishing an appset.
- The FONTS folder contains the fonts installed by Word. Leave this as is.
- HISTORY contains more files used by Internet Explorer. Remove this folder completely.
- INSTALLDIR corresponds to the installation folder created with the Publisher. It contains the main program files are used by Word. Do not make any exclusions or removals on its contents. The only task you must do here is to set access tokens for .exe files (.dll files can also be set with access tokens).
- INTERNET_CACHE corresponds to the Temporary Internet Files folder in the publishing machine. Remove this folder completely.
- O: corresponds to the Publishing drive. This folder is never needed and should be removed. All of the vital files in this drive are already included in the INSTALLDIR folder.
- The PROFILE folder contains files and folders with personal preferences and application-configuration information. Remove this folder so that you do not supersede a client machine's corresponding folder.

- PROGRAM_FILES contains folders that should be left alone and ones that can be safely removed.

Common Files is an empty folder, and will not be used by Word. Important files in the actual Common Files folder in the Publishing machine will be included in another special folder named PROGRAM_FILES_COMMON. This folder should be removed.

Wyse contains files and folders that were modified during the Publishing process. Traces of the Publisher should be removed from all appsets. Remove this folder from the build.

Microsoft Frontpage corresponds to the "Program Files\microsoft frontpage" folder on the publishing machine. It contains Frontpage server administrator files and snap-ins. This folder is used by Office 2003 and should be left as is.

Microsoft Visual Studio contains common files and folders used for debugging and development purposes for Office. This folder should not be modified in any way.

OfficeUpdate11 is a folder generated when downloading updates for Word through Microsoft's Web site. It contains temporary resources that were used only in the installation of any updates. You can remove this folder completely.

WindowsUpdate is generated similarly to the OfficeUpdate11 folder above. This folder is used only for patching purposes during the installation and should be removed.

- All exclusions and removals for the folder are now complete. Recall that you must set access tokens for at least all the .exe files within PROGRAM_FILES.
- PROGRAM_FILES_COMMON corresponds to the Program Files\Common Files folder in the Publishing machine. It contains Microsoft Office files used by Word and the Office Tools that were installed. Setting access tokens is all that should be done here.
- PROGRAMS is an empty folder You can remove it and its subfolder.
- STARTMENU is another empty folder. Remove it.
- SYSTEM contains some files and folders that should be removed and others that should be left alone depending on the application being published. This folder corresponds to the System32 folder in the Publishing machine.
 - Catroot2 is a system-specific folder. Windows XP re-creates it at every reboot. It does not contain any pertinent files to Microsoft Word; therefore, you should remove it. Leaving it in may corrupt a client machine's corresponding folder.
 - Config contains files that store system and application settings. These are machine-specific; therefore, you should remove them to avoid causing a client machine to malfunction.
 - Ias is the Windows Internet Authentication Service folder. There are no files essential to Word; therefore remove this folder.
 - Inetsrv contains a subfolder called MetaBack. Windows stores backups of the Windows IIS Metabase in this folder. The metabase is used by Internet Explorer for fast retrieval of IIS configuration information. It is not needed for Microsoft Word; therefore, remove it.
 - Spool is a folder used by Windows to store printer configurations. Some applications may need to install printer drivers or other files in this folder. This has no relevance to Word so you should remove it.
 - Wbem contains log files used by Windows Management Instrumentation. This should be removed.
- The SYSTEM folder itself contains numerous .dll and .exe files that are vital for Word to function properly. Set the access tokens for these files.

- SYSTEMDRIVE corresponds to the root drive of the publishing machine (C:). It does not contain anything that would affect Word so you can remove it from the build.
- The WINDOWS folder contains sensitive files and folders, some of which are not needed for Word.

Downloaded Program Files contains any office update engine files that were used in updating Microsoft Word. Remove this folder.

Fonts is an empty folder. Microsoft Word's installed fonts have already been included in the special FONTS folder that we've covered previously. Remove the folder.

Help contains a file named hicolreg.dat that is used by Microsoft Visual Studio for help documentation. This file was created because Word has a Script Editor. Leave this folder as is.

Installer contains very important .exe and the local .msi file(s) created by Word's installation. Nothing should be removed from this folder, as doing so may corrupt functionality for the application.

Msapps is a shared folder among all Microsoft applications. It was empty when Word was installed. Unless it contains a shared resource used by Microsoft Office, you should remove it.

Prefetch contains system specific prefetch files. These files are generated by applications that are executed within the Publishing machine. Remove this folder.

ShellNew contains a list of template files for Windows. When you right-click and select **New**, you are presented with a list of files you can create. This folder contains templates for both Binder and Word files. It should not be removed.

System contains no files and can be safely removed.

System32 has basically the same use as the above folder. The contents of this folder are already included in the special SYSTEM folder that we covered previously. This folder can be safely removed.

Temp contains temporary files used by Windows. It should be removed.

Twain32 is where Windows stores any files and drivers for digital cameras and scanners. It can be removed because Word does not use it.

- The WINDOWS folder contains several files you can remove. All the log files can be removed because they are regenerated by Windows. Leave all other files in this folder as is. Set the access tokens for the entire folder.

This completes clean-up of the Files section of the build file. Stop recording the macro and save it for use later.

Cleaning Registry Settings in the .aeb File

The registry entries being removed or excluded have been categorized under one of the major rootkeys that are caught when publishing.

HKEY_CURRENT_USER

- Software\Wyse contains registry keys related to WSM Publisher. No traces of the Publisher should be included in any appset being published. This key should always be removed.
- Software\Microsoft contains several registry entries that may or may not be necessary for publishing.

- Windows\CurrentVersion\Explorer contains registry entries related to recent Windows Explorer activity. Most entries here can be removed from the .aeb file. However, there are keys which may be needed for the application being published.

ComDlg32 stores information on opened and saved files that were recorded by the OS during the publishing process. This should be removed.

FileExt contains any new file extensions that were created during the publishing process. Remove any other file extension not related to the application being published.

MenuOrder maintains start menu shortcut on the publishing machine. Removing this will have no effect on the application being published.

RunMRU contains a list of recent commands executed through the Run dialog box on the publishing machine. This should be removed.

Streams stores information about the size and location of closed Windows used during the publishing process. This also must be removed.

UserAssist contains user assistant history files. This can be safely removed for any application being published.

- Windows\CurrentVersion\Extensions contains information on what program the OS should use to open certain file extensions. The extensions found here are usually related to the application being published (in our case, Word). Therefore, do not remove anything from this key.
- Windows\CurrentVersion\Internet Settings contains local Internet Explorer settings specific to the publishing machine. Remove this key.
- Windows\CurrentVersion\Shell Extensions contains system-specific shell extensions for Windows Explorer. Remove this from the .aeb file.
- Windows\Shell contains information on any desktop activity during the publishing process. Remove this key.
- Windows\ShellNoRoam contains information regarding recently run programs and their windows sizes and positions. Remove this to prevent overriding a local client machine's own settings.
- Netscape contains Word 2003 plug-ins for Netscape Navigator. Do not remove these.
- ODBC contains database information related to Word and its additional tools (such as Access and Excel). This key should be left alone.

HKEY_LOCAL_MACHINE

- Software\Classes contains information about an application that is needed to support COM functionality. Word is an msi based application; thus, it generates a single key that should merit our concern.
- Installer contains information regarding the Word 2003 installation. It is advised that all references to the installation source of Word be removed from within this subkey.
- Software\Clients contains a file association linking Word to Outlook envelope files. This key should be left as is.
- Software\Microsoft contains several keys that should be removed (described below).

Cryptography stores local digital signature settings specific to the publishing machine. This can be removed.

Windows contains registry keys that should be removed, including:

- Installer\UserData holds more installer information for Word. Remove the install properties key named InstallSource. You do not want the application looking for an installation source that is specific to the publishing machine only.
- ModuleUsage contains files related to the Office update installation engine. This key will be present if Word was updated from Microsoft's site before the second snapshot was taken. This key can be removed from the .aeb file.

- OfficeUpdate stores data transfer information related to the Office update engine. This should be removed also.
- Uninstall stores information that Word will need to uninstall itself from the publishing machine. The information here is system specific. This entire key should be excluded from the appset.
- System\ControlSet001 is basically a duplicate of CurrentControlSet. This entire key can be removed safely from the .aeb file.
- System\CurrentControlSet stores any system settings and services that may have been installed with Word. In our case, this key holds no essential information for Word. Therefore, the entire System key can be removed completely.

HKEY_USERS

This registry rootkey stores user-specific settings for the publishing machine. Information stored here is not necessary for Word to function properly. The entire HKEY_USERS key can be removed completely from the.aeb file.

Conclusion

Now that clean-up is complete, the build file can be saved. This is the file that will be used to create the appset.

- After saving, select Create application set from the File menu. You can then create a new config file or load a config file.

In this case study we saved our configuration file when we created our installation folder, so we can use that. If for some reason you needed to generate a new config file, you would have to use the GUID associated with the application.

- From this point, you can specify additional information about the appset to be created. You can optimize the appset with compression, encryption, and the inclusion of a specific prefetch file when recreating the appset for republishing. The Publisher will create the appset after you enter the required information in the wizard.

This page intentionally blank.



A

Handling AppEvents

This chapter provides important information on handling AppEvents.

To convert a locally installed application into an application capable of being streamed on a network, the Publisher takes a snapshot of the states of the installed application and records them in the appset file. When an application is activated by the user, those states are re-created on the user's system. Sometimes, however, the re-created static states of the application do not allow it to run. In these cases, it may be necessary to adjust those states or the run-time environment before, during, or after activation, to allow the application to run properly. In other words, it may be necessary to automate certain actions that cannot be captured in an appset.

For example, it may be necessary to configure all the plug-ins that the application provides for office suites and browser applications. It may also be necessary to configure virtual printers that the application requires. Another example is a video game that runs a video performance application during the installation process to configure the game parameters. It may also be necessary to start a program in the background that should always be running.

To allow such adjustments to be made dynamically, for example upon activation of the application, WSM Client provides an event-driven framework that can be used to trigger the execution of custom code in response to a variety of events called *AppEvents*. The custom code that is used to handle a particular AppEvent is called an *AppEvent handler*. These AppEvent handlers are executed by a special dynamically linked library (DLL) called Custom AppEvent DLL (CAED) that must be included in the appset during the publishing process. The CAED must be configured to execute the appropriate AppEvent handler for a given AppEvent. This can be done either through a provided GUI-based Publisher macro or directly in the user's system registry.

The CAED does not contain the AppEvent handlers. Instead, this custom code is implemented in a separate module that the CAED can call. This provides great flexibility to customers because it allows them to develop AppEvent handlers in their programming language of choice, thereby leveraging available programming skills and perhaps even existing code.

AppEvent Types

Table 4 lists the various types of AppEvents that WSM Client sends out to the CAED and indicates when each AppEvent is sent out.

Table 4 AppEvents and descriptions

AppEvent	Description
ClientStartup	Sent as the client is starting up.
ClientExit	Sent as the client is shutting down.
PreInstall	Sent immediately prior to an appset being activated.
PostInstall	Sent immediately after an appset has been activated.
PostAppStart	Sent after the application starts (if an access token has been set for the executable).
PostAppExit	Sent after the application exits (if an access token has been set for the executable).
PreUninstall	Sent immediately prior to an appset being deactivated.
PostUninstall	Sent immediately after an appset has been deactivated.

AppEvent Handlers

When one of the above events occurs, the CAED will execute the corresponding event handler, which is configured by the customer either during the publishing process or directly on the client machine.

Event handlers may be configured with a command line (executable and parameters), working directory and blocking time-out (how long to wait for the handler to complete before returning control back to the client), and these settings are stored in the registry.

A WSM Publisher macro is available to aid in the configuration of event handlers by setting-up the registry appropriately.

For most AppEvents, it is recommended that the executables and registry settings associated with the AppEvent handlers be included in the appset during the publishing process. In this way, they become automatically available once the appset has been activated, and up until deactivation takes place.

This approach will work fine for the following events: ClientStartup, ClientExit, PostInstall, PostAppStart, PostAppExit, PreUninstall. However, it will not work for the PreInstall and PostUninstall events. Indeed, because these two events are fired at a time when the appset is not yet, or is no longer, activated, the executables and configuration settings associated with their handlers will not be available if located in the appset. To handle these two events, the executables and registry settings must be made available on the client machine by some other means prior to application activation, and should still be available immediately after deactivation.

Note that even if a handler's executable is local to the user's machine, the event handling mechanism will not work unless the handler's configuration settings are also present on the user's machine at the time of activation because, without these settings, there would be no way for the CAED to determine what handler to dispatch the event to.

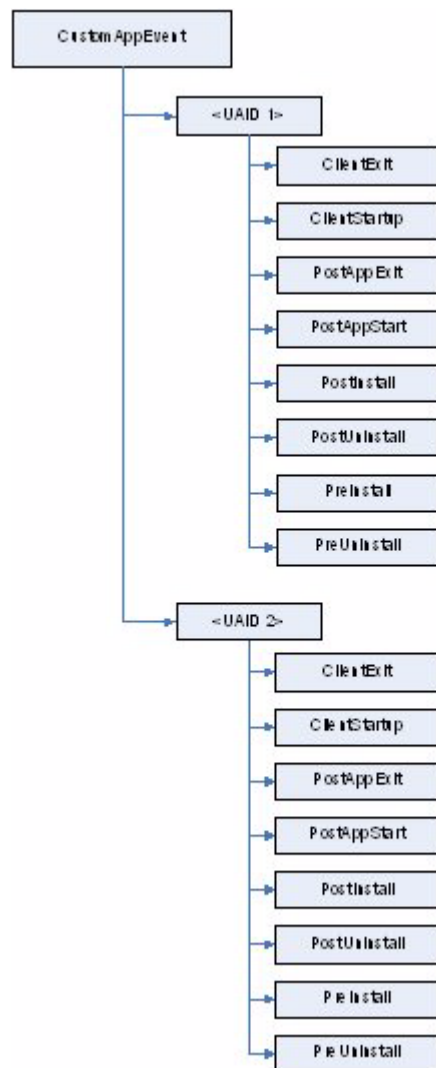
Also note that only one handler may be configured per AppEvent. Additionally, AppEvent handlers have no control over client actions. So, for example, it would not be possible for a handler to prevent an appset from being started, activated, or deactivated.

Handler Configuration

Handlers are configured in the registry in a structure that is simple and makes it easy to customize the various handler properties.

The basic registry structure is shown in the following figure and is located directly under registry key HKEY_CURRENT_USER\Software\Wyse\Wyse WSM:

Figure 34 Registry Structure



This diagram depicts the hierarchy of registry keys that are used by the Custom AppEvent framework. Each appset that has been configured with Custom AppEvent handlers will have a key under the CustomAppEvent root key named according to the unique application ID (UAID), which is a GUID. Each appset can have one handler for each of the defined events, as shown in Figure 34. Each handler key will have several values corresponding to the configuration settings mentioned earlier (see Table 5).

Table 5 Value names and types - requirements and descriptions

Value Name	Type	Required or Optional	Description
LaunchCmd	REG_SZ	Required	<p>Command line to execute.</p> <p>The string must include the module executable and, optionally, command line arguments. The module name must be enclosed in double quotes if it contains spaces to indicate where the file name ends and the arguments begin.</p>
WorkingDir	REG_SZ	Optional	<p>Initial drive and directory for the handler.</p> <p>The string must be a full path that includes a drive letter. If omitted, no working directory may be assumed.</p>
Blocking	REG_DWORD	Optional	<p>Amount of time (in seconds) to wait for the process to complete.</p> <p>The maximum possible value is 5 minutes (300 seconds). If omitted or if set to zero, no blocking is performed.</p>

As indicated earlier, it is possible to configure handlers to execute from files that are included within the appset itself. Paths for special folders may be specified using their symbolic representation instead of the actual folder path. For example:

[OT_APPDIR] = O:\{appid}

[OT_INSTALLDIR] = O:\{appid}\INSTALLDIR

[OT_SYSTEMDRIVE] = C:

[OT_WINDOWS] = C:\Windows

[OT_SYSTEM] = C:\Windows\System32\

**Note**

WSM Publisher will automatically translate instances of the installation folder path to the appropriate symbolic representation (that is, [OT_INSTALLDIR]).

Configuration Macro

To assist publishers in configuring handlers during the publishing process, a Publisher macro (CustomAppEvent.aem) is provided that prompts the user for the information necessary to configure the handler registry keys.

The publisher is initially required to provide the application ID for the appset being published. The publisher is then asked to provide values for each handler as defined in the previous section. All data supplied by the publisher is added to the registry keys in the appset and are subsequently created on the client machine when appsets are activated.

Environment Variables

It is sometimes necessary for a handler to have some information about the running application or the current user in order to accomplish its task. The CAED provides such information to every handler that it calls through environment variables (see Table 6).

Table 6 Environment Variable names and descriptions

Environment Variable Name	Description/Comments
STREAMING_INSTALLDIR	Folder where the application is installed.
STREAMING_USERNAME	User name of the currently logged on user.
STREAMING_USERDOMAIN	Domain under which the current user is logged.
STREAMING_UAID	Application ID – a GUID that uniquely identifies the application to WSM.
STREAMING_EVENT	Name of the AppEvent. See "AppEvent Types."



Note

Handlers are executed as the current user, and in their own process.

Handling AppEvent Example

This section presents a very basic example to illustrate the handling of an AppEvent.

Suppose that we want to have end users log what they intend to do every time they run a certain WSM-based application and that the log entries must be made in a text file named with their user name.

One way to accomplish this is to set the LaunchCmd registry value to notepad.exe "C:%STREAMING_USERNAME%.txt" in the PostAppStart key located under our application's key (that is, the one named with the corresponding UAID).

This will now cause the desired log file to open in Notepad every time our application starts. If the file does not exist, Notepad will create it after asking the user for confirmation.

Of course, in reality there are more appropriate tools to accomplish the proposed goal, but this example was based on Notepad for the sake of simplicity, and because it is easy enough to try out.

Publishing Instructions

This sections provides instructions that one must follow, beyond the normal publishing steps, in order to use the CAED.

Including Handler Executables in the Appset

To include handler executables in the appset, simply insert the corresponding files in the application build before creating the appset with the Insert File command.

**Note**

If you want to handle the PreInstall or PostUninstall events, it will be necessary to make the executables available on the client machines independently of the appset before the latter is activated.

Setting-up the Registry Configuration

To configure the handler settings in the registry, you should run the CustomAppEvent.aem macro after the build has been created, and before you create the appset.

Alternatively, you may enter the appropriate registry settings manually.

Adding the CAED to the Appset

To add the CAED to the appset, be sure to specify the CustomAppEvent.dll file in the AppEvent DLL field in the Advanced page of the Appset Creation Wizard.

Figures

1	Workflow	4
2	Publishing steps - docked	5
3	Publishing steps - undocked	5
4	Create Installation Folder	15
5	Create Snapshot - pre-installation	17
6	Creating Snapshot	17
7	Create Snapshot - post-installation	19
8	Create Build from Diff of Two Snapshot Files	19
9	Creating Diff	20
10	Done Creating Diff	20
11	Publisher Build File window	22
12	Play Macros	24
13	Publisher	26
14	Insert Item	29
15	Insert Item - registry root	30
16	Insert Item - select subfolder	30
17	Insert Item - select subkey	31
18	Edit String Value	32
19	Modify Registry Value Type	33
20	Record Macro	33
21	Remove Folder	34
22	Find	35
23	Toggle Access Token For Files	35
24	Edit Folder Disposition	37
25	Create an Application Set wizard	39
26	Main Options	40
27	Authoring Information	41
28	Supported Operating Systems	42
29	Advanced Options	43
30	Summary	45
31	Creating Application Set	45
32	Appset Viewer	47
33	Macro Editor	60
34	Registry Structure	89

Tables

1	WSM Publisher File Types	8
2	Generic Publisher Macros	25
3	Insert, Remove, Include, and Exclude operations	29
4	AppEvents and descriptions	88
5	Value names and types - requirements and descriptions	90
6	Environment Variable names and descriptions	91

Publisher Guide

Wyse WSM™ Release 3.0
Issue: 011310

Written and published by:
Wyse Technology Inc., January 2010

Created using FrameMaker® and Acrobat®