

The Basics of Application Packaging:

Best Practices for Enabling Reduced Software Management Costs

Application packaging can help enterprises manage growing volumes of software for desktop and server systems efficiently. By streamlining software configuration and deployment, application packaging can help reduce application management costs. The information in this article pertains to OS migrations for desktop systems, including best practices for implementing application packaging techniques.

BY JUKKA KOULETSIS

Related Categories:

Altiris/Wise

Altiris

Application development

Application packaging

Microsoft Systems
Management Server (SMS)

Systems management

Visit www.dell.com/powersolutions
for the complete category index.

Maintaining desktop and notebook systems has become an expensive proposition for many corporate networks. New application management techniques are being developed to help enterprises administer their existing PC investments more efficiently, reduce end-user support costs, and minimize end-user business disruptions. One such development is application packaging, which is emerging as a critical component of an overall software configuration management strategy.

Application packaging involves the preparation of standard, structured software installations targeted for automated deployment. Automated installations, or *packages*, must meet the installation requirements for a specific environment: corporate standards for software usage and desktop design, multiple languages, regional issues, and software-related support issues. In addition, packages must be prepared for both commercial software and applications developed in-house.

To enable this level of application management, Microsoft now provides the Microsoft® Windows® Installer (MSI) service as a part of its desktop operating systems. Starting with Microsoft Windows 2000, the Windows Installer has been included in the base desktop OS. For earlier OS versions, Windows Installer can be downloaded from Microsoft's Web site. This database-driven service

resides on workstations and controls the installing, uninstalling, patching, and repairing of software. Input into the Windows Installer is an .msi formatted file, which is further explained in the "Understanding Microsoft Windows Installer" section in this article.

When deploying new applications, many administrators consider an automated approach to software installation as an attractive option to help save time and help reduce compatibility issues. However, the testing needed to verify that a new application will not cause an old application to fail can be difficult, if not impossible, in an uncontrolled environment. Although business disruptions caused by a new application deployment often cannot be measured directly, impaired end-user productivity can be costly. This article examines how application packaging practices—and in particular, Windows Installer—can help address common application management concerns.

Understanding Microsoft Windows Installer

The Windows Installer service was designed to support every phase of the application management life cycle, providing a service to support each step involved in managing a desktop application from deployment through retirement (Figure 1). To support these functions, the Windows Installer needs to receive instructions from an

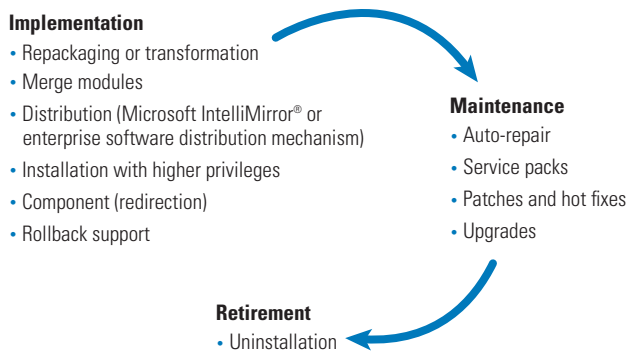


Figure 1. Application life-cycle tasks supported by Windows Installer

installation package. Previously, installation packages took the form of a setup.exe file. Unfortunately, inconsistencies in the way independent software vendors and internal software development groups created these installation files sometimes led to complications when administrators attempted to manage automated installations.

The emerging standard is for Windows Installer to use the msixec.exe program to process the installation packages at an end user's PC. The packages follow a standardized database structure containing the information that Windows Installer requires to install or uninstall an application and to run the user interface for the setup. Each installation package includes an .msi file containing the installation database, a summary information stream, and data streams for various parts of the installation. The .msi file can also contain one or more transforms, internal source files, and external source files or cabinet files required by the installation. Figure 2 shows Windows Installer file extensions.

This approach enables Windows Installer to determine components that belong to an application, and to safely remove application components and restore a system to a working state. Furthermore, because Windows Installer is a service, it is designed to support software installations as the local Administrator role in locked environments, enhancing the application process.

Windows Installer can support applications installed from a network share—referred to as an administrative installation—or locally on an end user's PC. The downside to using a network share can be that systems receive patches or repairs only when they are connected to the network, which may be a consideration for organizations supporting many notebook users.

Creating MSI files

To build the installation package in the correct MSI database format, developers must collect information about each application. Items include executable files, installation instructions, configuration parameters, test instructions, and hardware and software dependencies. Once this information is gathered, the packaging effort is designed to produce a Windows Installer–based

installation package for each application, including the installable MSI as well as any deployment files and wrappers required to distribute the package electronically to a PC.

Best practices recommend that installation packages be created by experienced packaging engineers, using tools specifically developed for that purpose. For example, Dell engineers have used Altiris® Wise Package Studio® software for efficient development of MSI installation packages. At minimum, the tools used to build installation packages should provide the following capabilities:

- OS certification analysis
- Capture of the installation snapshot
- Custom scripting and the creation of transforms
- Checking for compliance with packaging standards
- Conflict resolution
- System and end-user testing
- Integration with various software distribution engines

A completed package file should conform to standards and successfully install, uninstall, update, and function on the targeted OS. One package must be produced for each unique application configuration. Successfully packaged applications can be delivered to the project test team in a format ready for deployment on software distribution engines such as Microsoft Systems Management Server (SMS) 2003, the Altiris Software Delivery Solution™ tool, or Computer Associates Unicenter. Any department-specific requirements for installation can be handled through the creation of transforms within an existing MSI file.

Quality assurance (QA) testing should be performed during the creation of installation packages to verify that the MSI database is operating correctly to install, uninstall, and repair an application. In addition, peer reviews help ensure a high first-pass yield through the application distribution process by not requiring special permissions each time an application is installed.

Conflict management is a key aspect of package creation. Because of the way in which dynamic-link libraries (DLLs) are built

Extension	Description
.msi	Windows Installer database
.msm	Windows Installer merge module
.msp	Windows Installer patch
.mst	Windows Installer transform
.idt	Exported Windows Installer database table
.cub	Validation module
.pcp	Windows Installer patch creation file

Source: Microsoft Developers Network, msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/windows_installer_file_extensions.asp.

Figure 2. File name extensions used in the Windows Installer service

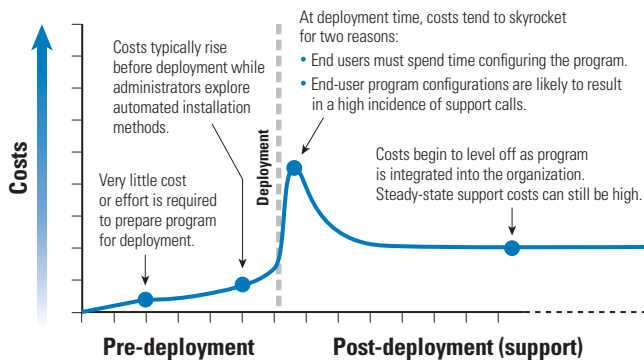


Figure 3. Deployment costs without an application packaging strategy

into a Microsoft OS, packaging engineers must determine whether a new application introduces conflicts. The conflict management process is designed to ensure that DLLs do not interfere with each other and that any required application isolation is configured into the MSI database.

Incorporating application packaging into software management

The application packaging process must fit within an organization's overall software configuration management strategy. Associated software configuration management tasks include application inventory and asset management, profile management, requirements gathering, user acceptance testing, and electronic software distribution.

Application inventory and asset management. This task involves compiling a list of applications that reside on each desktop system, grouping the applications by line of business, and determining which applications are necessary. Subsequently, administrators may load inventory information into a database that tracks assets and licenses on an ongoing basis.

Profile management. In this task, applications are logically grouped into *bundles* and entered into a profile management system. Each bundle is created within the context of a tier—for example, tier 0 is for the base OS, tier 1 is for company-wide applications, tier 2 is for division-wide applications, tier 3 is for departmental applications, and tier 4 is for individual user applications.

Requirements gathering. This task gathers the requisite technical data about each application, including the installation source code, test scripts, installation and uninstallation instructions, and any configuration details.

User acceptance testing. During this task, testing performed on an application package is designed to ensure that the application is functioning properly and to avoid business disruptions caused by application failure.

Electronic software distribution. This task involves moving the application package onto the software distribution server, performing

QA to help ensure accurate delivery to a PC, and transitioning into steady-state support.

Minimizing application management costs

By implementing an application packaging strategy, organizations can help reduce administrative costs while providing business benefits. For example, this approach enables administrators to set and enforce corporate software configuration standards and to minimize the frequency of administrative errors during installation. The Windows Installer service also offers features that help streamline application deployment, including powerful self-repair and rollback capabilities that are designed to dramatically reduce the occurrence of deployment-related desktop software problems.

Figures 3 and 4 compare generalized application deployment scenarios. The Figure 3 example shows a typical pattern of deployment costs that are incurred when application packaging is not used, while the Figure 4 example shows a typical pattern of deployment costs when application packaging is used. *Note:* These are general findings; actual figures would be based on an organization's specific software configuration management cost structure.

Adhering to best practices for application packaging

To realize the benefits of application packaging, IT organizations should adhere to best practices for the following activities: gathering requirements, using Windows Installer, building a stable core image, managing conflicts, evaluating application suitability for packaging, establishing a centralized packaging process, creating a structure to group applications, and implementing a formal request process.

Requirements gathering. Those responsible for creating the application packages must collect the needed details about each application. Best practices recommend that an engineer experienced in creating MSI packages be responsible for gathering the technical data about an application.

Use of Windows Installer. Microsoft's Windows Installer technology is designed to simplify the process of adding applications to

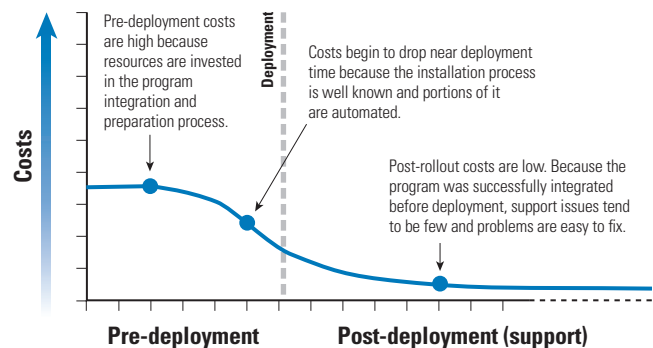


Figure 4. Deployment costs with an application packaging strategy

STREAMLINING APPLICATION PACKAGING WITH WISE PACKAGE STUDIO

Wise Package Studio is an application life-cycle management solution used by deployment and desktop management teams to prepare applications for the enterprise. Based on a structured application management, packaging, and QA process known as enterprise software packaging, Wise Package Studio helps administrators migrate to MSI-compatible application and patch packages while enabling high-quality, reliable deployments that support corporate standards. In turn, organizations can benefit from quick software rollouts, streamlined Windows Installer migration, and high return on Windows 2000 and Windows XP investments. The Wise Package Studio product family includes Professional Edition, Quality Assurance, Enterprise Management Server, and Standard Edition.

Wise Package Studio also offers advanced integration with Altiris IT life-cycle management products, including Altiris Patch Management Solution™ software. For more information, visit www.wise.com or www.altiris.com.

the desktop and to minimize support costs by helping to eliminate errors associated with those installations.

Stable core image. Engineers should not begin packaging until the core OS image has been defined and stabilized. Attempts to build and test packages on early versions of a core image typically lead to significant reworking of the packages on the final core image.

Conflict management. Even when IT organizations identify conflicts between old and new applications, they do not always have a structured method for resolving the conflicts. Packaging tools can help to quickly identify common conflicts and then aid in establishing policies for conflict resolution as well as for automating the conflict-management process.

Application suitability for packaging. Although application packaging offers several benefits, certain applications should be deployed outside an automated software distribution. Rarely is an installation package created for every application that is deployed across an organization.

Centralized packaging process. Although enterprises typically use numerous Windows-based applications, many enterprise IT organizations do not have a common organization or methodology for software packaging and deployment. Selection of a packaging tool set can facilitate this centralization.

Structured application grouping. While some applications are used across an organization, others may be useful only to a certain business unit, geographic location, or group of specialized users. By instituting a packaging process, IT organizations can establish

a structure for classifying and managing diverse software used throughout a large enterprise. Typical groupings include applications core to the business, those used primarily for departments or business units, and those deployed ad hoc to small groups of users.

Formal application request and approval process. Packaging technology can help organizations implement a formal process for requesting, approving, and distributing applications. A simple workflow process can provide a way for users to request a specific application and obtain approval from the appropriate business and IT managers. This process can significantly aid in eliminating unnecessary application deployments while helping to ensure that end users receive the appropriate, predefined versions of the requested applications. This approach can lead to a reduction in the cost and complexity of support services.

Achieving efficient, cost-effective application deployment and software management

Application packaging can be an important component for efficiently managing the increased volume of software on desktop and notebook systems. By streamlining software installation, uninstallation, repair, and patching, application packaging can help reduce costs associated with each phase of the application deployment and support life cycle. In particular, application packaging is designed to reduce costs and improve efficiency during the deployment and post-deployment phases. Such benefits depend on having a stable environment from which to automatically distribute packages to PCs, using tools such as Microsoft SMS, Altiris Client Management Suite™ software, and Novell® ZENworks® software.

By enabling fast, standardized software installations, application packaging is designed to minimize desk-side visits by support staff and avoid business disruptions caused by software failure—thereby helping to reduce costs for IT support and business operations, respectively. When implemented as an IT best practice, application packaging can help create a cost-effective software repository that is in line with overall business priorities. ☞

Jukka Kouletsis is a senior consultant with Dell Services. He has been involved in technology migrations for more than 20 years, including seven years with Dell. Jukka has a B.S. in Computer Science from Hofstra University in New York.

FOR MORE INFORMATION

Wise Resource Center:

www.wise.com/rescenter.asp

Microsoft Windows Installer:

msdn.microsoft.com/library/default.asp?url=/library/en-us/msi/setup/roadmap_to_windows_installer_documentation.asp