U3 Deployment Kit

Version 1.0
May 2006
Revision 1.0

U3
303 Twin Dolphin Drive, Suite 200
Redwood City, CA 94065
1-800-837-3654, info@u3.com

For more information, visit
www.u3.com

Return to Overview

U3 Deployment Kit Overview

1. Introduction

Welcome to the U3 Deployment Kit. For many developers, the documents in this Kit are all you will need to develop a U3 smart application. The contents of the U3 Deployment Kit are detailed below. You can click on the various items to link directly to the information you're looking for.

2. Overview

The process for creating a U3 smart application is as follows:

2.1. Step One: Develop

2.1.1. Profile Your Application

- **Start with:** U3 Application Types
- **Read:** Migration Analysis, part of the Application Deployment Guide
- **Read:** Integration Factors Worksheet
- **Read:** When to Use the DAPI

2.1.2. Scope Your Application

- **Read:** Application Deployment Guide
- **Preview:** U3 Smart Application Certification Overview and Self-Test
- **Read:** Developer Guide in the U3 SDK

2.1.3. Develop Your Application

- **Use:** Application Deployment Guide
- **Refer To:** Application Deployment Guide Quick Reference Guide
- **Use:** U3 Manifest Creator User Guide
- **Use:** U3Action User Guide
- **Use:** U3 Knowledge Base
- **Use:** If using the DAPI, download the U3 SDK. Start with the Developer Guide

2.2. Step Two: Get Certified

- **Use:** U3 Smart Application Certification Overview, Self-Test, and Self-Test Tools

2.3. Step Three: Go-To-Market

- **Use:** U3 Application Certification Overview, Self-Test, and Self-Test Tools
- **Use:** Go-To-Market Toolkit
- **Apply For:** U3 Software Central

3. Key U3 Resources

U3 Knowledge Base

U3 email support

Developer Forum Resources

To make sure you have the latest version of this document, please click here.

Return to Overview

U3 Application Types

1. Overview of U3 Applications

There are three types of U3 applications that can be developed and deployed.

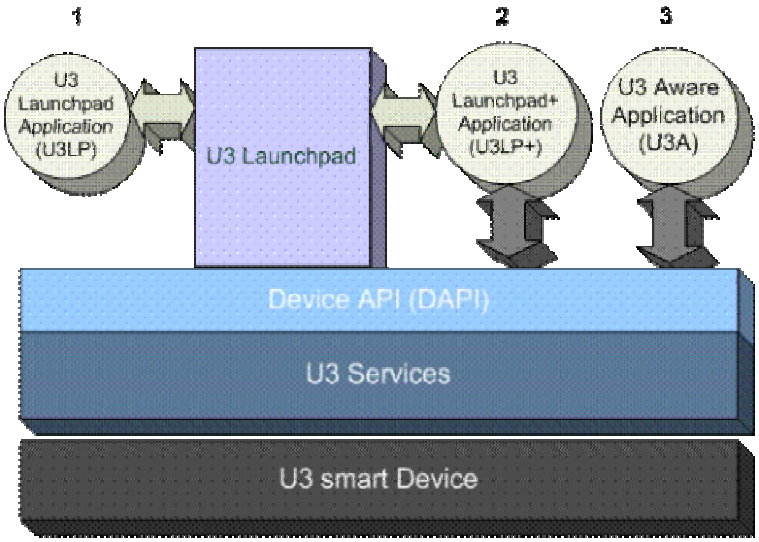


Figure 1: U3 Application Types and SDK Architecture

2. U3 Launchpad (U3LP) Application

A Windows application that is resident on the U3 smart device and whose lifecycle is managed by U3 Launchpad. Because the application is installed onto the device, all configurations, user preferences and files travel with the device. For further information and a description of the U3 Launchpad, refer to the *Application Deployment Guide*.

3. U3 Launchpad+ (U3LP+) Application

A U3 Launchpad application that ships with the DAPI DLL allowing it to communicate directly with the U3 device and leverage the advanced device features. This is necessary for example, to work with cookies, or to establish the device unique ID. For further information, refer to the *Application Deployment Guide*, *SDK Developer Guide* and the *DAPI Reference Guide*.

4. U3 Aware (U3A) Application

A host-based application that uses the DAPI DLL to detect and communicate with U3 Devices as they are inserted into the host. A U3A application is a standard Windows application, with a standard installation and startup procedure, which has been extended to

Return to Overview

U3 Application Types

support U3 devices. For further information, refer to the SDK *Developer Guide* and the *DAPI Reference Guide*. Examples of U3 Aware applications are:

- A Windows service, such as a Login service, that uses U3 smart device as tokens.
- A Windows application, such as a Backup tool, which has U3 device support, enabling data to be securely backed up to a password protected private area on the device.

Return to Overview

U3 Migration Analysis

1. Migration Analysis

This section discusses some of the U3 application requirements and recommendations introduced in Section 4.4 of the Application Deployment Guide. Implementation considerations are described in detail to demonstrate architectural thinking behind U3 applications.

1.1. DLLs and libraries required by applications that are not part of the minimum installation of the supported operating systems must be packaged with the U3 application files.

If the application relies on MFC DLLs that are copied to the host at install time if they do not exist, a mobile version would require that these be installed every time the U3 device is inserted into a new host machine. Implementation options are as follows:

- Copy the files to the system32 directory the first time the application is run on a new host machine. This may be problematic as U3 requires that the application remove all system changes when it stops running, and the files may be in use by another application. In addition, copying files to the system32 directory may require administrator rights.
- Place the DLLs in the same directory as the application.
- Statically link the MFC.

1.2. U3 does not protect one application from editing a second application's configuration data. Each U3 application is responsible for testing the consistency and correctness of its configuration data.

Let's say that the application currently uses the unique directories and registry entries created by Windows for the **current user** to ensure that data files for each user are kept separately.

There may be multiple U3 devices inserted to the host machine at the same time with the same application installed. Therefore, the logged-in user account that is now common to all inserted U3 devices is no longer a valid method for separating two different sets of user and application data.

Care must be taken to ensure that if any data (either file or registry based) is written by the application to the host machine, identification information should be included to ensure that the data is not accidentally used by another instance of the same application.

Implementation options are as follows:

- When writing data to files or the registry on the host machine, add a device identifier into paths, e.g. \[device id]\
- Do not write data to the host machine, but store all data on the U3 device

Return to Overview

U3 Migration Analysis

1.3. Where possible, paths in configuration files should be relative to the U3_* paths defined in the Application Deployment Guide.

The paths to the current set of directories used by the non-U3 version of the application is obtained either from the registry or via Windows environment variables. The U3 application will use the U3 variables in their place to determine the location of each U3 directory that it will use to read and write files.

1.4. Files should be stored in the recommended directories.

As the application is now mobile, any data that must remain with the U3 application when the U3 device is moved from one host machine to another should be stored on the U3 device.

The U3 practice is for the application to prompt the user to store these files on the U3 device, ideally in sub directory on the U3 device Documents directory.

Section 6.7 in the Application Deployment Guide, Applying the Worked Example, describes the file placement strategy in detail.

1.5. When explicitly loading a DLL or using any other files to maintain application isolation, absolute paths should be calculated to ensure that the U3 application is loading the expected file.

An application should ensure that it loaded the DLLs that sit with the executable by including the full path in the DLL name.

1.6. Applications should minimize the amount of time that file handles are held open, particularly those of files on the U3 device.

This will not be an issue if the application does not hold any files open.

1.7. Use of the Windows registry should be avoided where possible. If this registry is used, the U3 application should undo any changes before closing.

If an application relies heavily on the registry, the data that is written to the registry must travel with the U3 application on the U3 device. Implementation options are as follows:

- Move all registry entries to a U3 device-based file and update the U3 application implementation.
- Continue to use the registry with minimal code changes in the main U3 application. The registry should be restored from the U3 device to the host machine before the application is executed, and copied back to the U3 device

Return to Overview

U3 Migration Analysis

- Continue to use the registry with minimal code changes in the main U3 application. The registry should be restored from the U3 device to the host machine before the application is executed, and copied back to the U3 device when the application stops. The data can also be written to the U3 device periodically to ensure that the U3 device copy is updated even if the U3 device is removed before the application can close and update the file in an orderly manner. As stated in requirement 2, if the registry is used, an additional element must be inserted into the registry key path to ensure that it is unique per U3 device, in addition to being unique to the current Windows user.

1.8. The developer should assume that the U3 application will be executed from a different temporary directory/drive every time it is run.

The application should use the U3 variables to ensure that it does not use any hard-coded paths to the U3 device in its configuration files.

Return to Overview

U3 Integration Factors Worksheet

This worksheet provides a guide to quickly determine the effort required to port an existing application or an application under development to the U3 platform. In general, the activities described here would need to be implemented even for applications that are already mobilized to run from a removable mass storage device such as a standard USB flash drive.

1. Standard integration tasks:

Summary:

A short list of the programmatic modifications and additional runtime activities that most developers will have to implement are summarized here:

- Modify application to use the U3 runtime environment variables for all file path operations, licensing, and runtime action behaviors. The result of this activity will be an application that is insulated from the dynamic factors resulting from being supported on a removable mass storage device. Note that advanced features provide by the U3 DAPI dll may be used in addition to the U3 runtime environment variables.
- Implement U3 Action executables to customize runtime phase behavior: application installation and uninstall, runtime host configuration if required, application start, application stop, and runtime host cleanup if required. The U3 Launchpad doesn't require a traditional installer application like InstallShield. The deviceInstall action is used to customize an application's installation.
- Develop multiple instance strategy (see IF #7 in next section)
- Incorporate U3 Upgrade Process for appropriate activities: Activation, upgrade, etc…
- Create Package contents: application manifest file, download manifest file, application icon file, U3 Software Central download page graphics, etc…
- Complete U3 Smart Application Certification Self-Test and submit application for compliance verification.

2. Integration Factors Checklist

Each integration factor in the following table should be reviewed to determine whether or not it applies to the application under consideration. Each integration factor refers to a separate section in this document that contains additional information as to how to evaluate the applicability of the integration factor itself and guidelines to resolve it.

For any integration factors that have been checked, the project manager should consult with the appropriate resources to determine the number of resources and time required for each factor's solution.

Return to Overview

Return to Overview

U3 Integration Factors Worksheet

Integration Factor	Yes	Section Link
Uses COM objects not part of minimum OS installation		IF #1
Application uses the registry		IF #2
Implements or uses a licensing/registration module		IF #3
Implements an upsell/activation feature: trial/freeware/shareware to payware version		IF #4
Implements an auto/manual updating feature		IF #5
Uses or implements a DRM		IF #6
Requires implementation of a multiple instance strategy	**X**	IF #7
Exceeds flash memory friendly file update rate		IF #8
Requires a review of software provisioning strategy.		IF #9
Requires a runtime file location strategy for addressing startup or other runtime performance issues unique to operating from a removable mass storage device.		IF #10
Application requires Admin Mode on Windows OS.		IF #11
Application written in Java		IF #12
Requires .NET framework runtime package		IF #13

The difference between a standard PC application and an application mobilized for the U3 platform is described in some detail in section 3 of the U3 Application Deployment Guide (ADG).

3. IF #1: Uses COM objects not part of minimum OS installation

Description:

Application uses COM objects. This is problematic as the objects would need to registered temporarily and then unregistered during the Host Clean Up runtime phase. If other applications attach to the object while it is registered, it can't be removed from the system thus violating one of the main precepts of the U3 value statement which is to leave the system configuration in the same state as it was before the U3 device was inserted.

Note that object registration requires Administrator permissions. The application would not be able to use these objects in a User Mode scenario which may be a common occurrence on public systems.

Solution:

- Stop using COM – convert to standard dlls that are placed in the same dir as the executable for example.
- Statically link if possible.

Return to Overview

U3 Integration Factors Worksheet

- If the level of effort to move away from COM object dependency is too high, it is possible to leave the basic application as a pc standard app and combine it with U3 functionality to make the both the licensing and migration of data between different installed version of the application itself mobile. A standard pc application can be made what is termed a “U3 Aware” application by incorporating the U3 Device API (DAPI). Using the DAPI in a standard pc application allows the application to easily do two things:
  1. Be aware of the presence of U3 device connect and disconnect events allowing it to use the U3 device as a licensing dongle.
  2. Integrate to a U3 device application that serves as a way to transfer data among different installed instances of the PC application.
- Use a package such as Thinstall to wrap the application in a stand alone executable.

4. IF #2: Application uses the registry

Description:

The application uses the registry. Registry usage can span from minimal where small static bits of data such as license keys or module tracking are placed there for reference or the usage can be dynamic where the registry is used to track program state from one runtime session to the next.

There are three principal issues that registry use causes.

1. One of the principle foundations of the U3 value is that the host machine configuration be left in the same state that it was in prior to the application running. Leaving the registry modified is a violation of both the intent and letter of the U3 certification criteria.
2. Use of the registry must be managed to be compatible with whatever the strategy will be for managing multiple instances of the program as per solution for **IF #7: Requires implementation of a multiple instance strategy**
3. Depending on the area of the registry accessed, Administrator privileges may be required to make runtime changes to the registry.

Solution:

- Move the data stored in the registry into a flat file format on the U3 device.
- Continue to use registry with configure and cleanup actions defined. Use a naming convention that guarantees that registry data is unique for each application instance. Have cleanup actions copy registry data to U3 device.
- If continuing to use the registry, develop a “physical eject” strategy:
- Inform user that they must “safe eject” to preserve their data to the U3 device for future use.
- Periodically save appropriate registry trees to device to minimize data loss in

U3 Integration Factors Worksheet

"physical eject" scenario.

- Use a package such as Thinstall to wrap the application in a stand alone executable.

5. IF #3: Implements or uses a licensing/registration module

Description:

The application uses a licensing module to protect appropriate versions of the application.

Solution:

It is necessary that the licensing module and process be modified for two reasons:

- **License tethering:** Modify the license to tether to the U3 device using either the U3 runtime environment variables or DAPI interface. Using the U3 runtime environment variables, a unique device identifier can be derive from the Vendor ID and the Device Serial Number. An alternative approach is to use the DAPI dll, a signed module, to access the low level unique Device ID.
- **Integrate to the U3 installation/upgrade process:** Appropriate actions will need to be developed in order to provide the appropriate end user interactions during application installation (presenting EULAs, having serial entered, etc...) and during subsequent upgrades. Please see the Application Deployment Guide for details on these runtime phases and the runtime environment support provided by the framework
- Consult U3 Developer Knowledge base: search for articles on "upgrade".

6. IF #4: Implements an upsell/activation feature: trial/freeware/shareware version

Description:

The application supports an upgrade licensing model that features some sort of trialware, shareware, or freeware version that can be upgraded through an upgrade mechanism.

Solution:

The upgrade process needs to be modified to work within the U3 upgrade process as documented in the ADG. Please refer to IF#3: Implements or uses a licensing/Registration module as the solutions applicable for that integration factor most likely apply here as well.

7. IF #5: Implements an upsell/activation feature

Description:

The application implements an auto/manual updating feature that allows the application to updated in the field. This process can be configured by the end user to be an automatic process or one where the user periodically requests if updates are available.

Solution:

The updating process needs to be modified to work within the U3 upgrade process as

U3 Integration Factors Worksheet

documented in the ADG. Please refer to IF#3: Implements or uses a licensing/Registration module as the solutions applicable for that integration factor most likely apply here as well.

8. IF #6: Uses or implements a DRM

Description:

The application uses a Digital Rights Management (DRM) technology to control or restrict use of assets. The assets requiring DRM protections are typically multi media assets.

NOTE: DRM techniques associated with preventing the reproduction and unrestricted distribution and use of the software application itself are covered in Section IF #3: Implements or uses a Licensing/Registration Module.

Solution:

For a U3 application, it is anticipated that the desire will be to tie or tether the DRM feature to the U3 device itself. There are two possible modifications to the DRM that might be required:

- **Tethering:**

  Modify the DRM to tether to the U3 device using either the U3 runtime environment variables or U3 Device API (DAPI) dll. Using the U3 runtime environment variables, a unique device identifier can be derived from the Vendor ID and the Device Serial Number. An alternative approach is to use the DAPI dll, a signed module, to access the low level unique Device ID.

- **Dynamic Path Support:**

  It may be necessary to modify the DRM so that it uses the runtime environment to change the DRM module's sense of where it is running from. It will have to default to locations on the U3 device itself unless host configuration changes are made during the host configuration runtime phase and then undone during the host cleanup runtime phase.

- **Use host machine resources:**

  During hostConfigure, check for required support (Windows Media Player 10 for example) before starting application. Discontinue startup if resources aren't not available and/or advise user to install required resources.

9. IF #7: Requires implementation of a multiple instance strategy

Description:

A strategy should be developed to handle cases where the same U3 application is executed from different U3 devices connected to the same host machine.

Note that the Launchpad itself doesn't restrict the number of times that an application can be launched from the same device.

This strategy should include handling the case where the corresponding standard pc version of the application is present on the host machine.

U3 Integration Factors Worksheet

Solution:

- Implement solution so that only one version of the application can run at a time.
- Implement solution where multiple instances are allowed to run simultaneously but each maintains unique copies of related data files.
- Implement solution where multiple instances are allowed to run simultaneously with arbitration scheme for allowing them to share appropriate resources.

10. IF #8: Exceeds flash memory friendly file update rate

Description:

NAND flash technology generally guarantees 110K program/erase cycles for each physical erase unit (the erase cycle limit may vary for each NAND flash device family). The implication here is that an application that frequently writes data to a flash device will impact the overall lifetime of the device. Higher quality flash drives implement wear leveling algorithms that increase the effective P/E cycle rate to the order of 10^6.

Solution:

Use the base P/E cycle rate of 110K for evaluation knowing that most U3 drives do implement wear leveling. If your application appears to pre-maturely wear out the flash memory (< 5 years), plan to have the frequently updated files temporarily copied to the hard drive for runtime activities. These files are then archived back to the U3 device for maintaining state during the host clean up runtime phase or cached writes can be periodically written.

P/E cycle rate calculator:

(Memory Capacity MB * PE Cycle Rating) / (MB Written per Sec * 3600 * hours application is used).

11. IF #9: Requires a review of software provisioning strategy.

Description:

There are scenarios where it may be required to restore an application that has been paid for but has been corrupted, lost, or the user wishes to transfer the license to another device. Most of these situations can be addressed by similar policies and in field solutions provided for corresponding standard pc applications. Some possible situations for which solutions should be considered are given here:

- U3 device is lost.
- User desires to transfer license and application to a higher capacity device.
- User has accidentally deleted the app from the device.
- Application has become unstable for some reason and requires re-installation.

U3 Integration Factors Worksheet

Solution:

Solutions for these situations may require that install and upgrade actions support the business rules used to address these situations in the field. Transferring a license to another device could be handled by a special installation executable that manages the license transfer as an example.

Explicit solutions are not prescribed here. The intent of this integration factor is to make the developer aware of the inevitability that one or more of these situations will occur. Most developers find that their existing business rules suffice with regard to this IF.

12. IF #10: Requires a runtime file location strategy for addressing startup or other runtime performance issues unique to operating from a removable mass storage device.

Description:

A developer can choose where their application executables and other application files are run from or stored accordingly. Section 6 of the ADG discusses the format of the U3P file structure and how the U3 Launchpad manages files based on where they are located in the U3P. Section 6 also includes some background, as well as an applied example.

The short summary is that the developer has a choice of having their application files (some or all) managed by the U3 Launchpad and copied to temporary locations on the host machine for execution. The developer can also elect to execute from files located on the U3 smart drive itself (U3 resident). U3 best practices do impose a few requirements on specific file locations as per the following note:

NOTE: All application stop and host cleanup executables must be stored in the host directory of the package regardless of where the application executable itself is run from.

Solution:

An explicit solution is not prescribed here. Here are some considerations to take into account:

A key metric is application startup time. Will the application run from the host machine or the device? Running from the host machine (i.e., the files are placed in the host directory of the U3 package file) means that the files will be copied to the host machine by the U3 Launchpad prior to execution. Running from the U3 device itself means that the application will essentially start instantly but there is the risk of instability in the physical eject scenario.

In general, provisioning data files and infrequently accessed runtime files to the device and core executables to the host is good approach.

IF #8: Exceeds flash memory friendly file update rate must be considered for U3 device based data files where the rate and amount of data written is high.

Return to Overview

U3 Integration Factors Worksheet

13. IF #11: Application requires Admin Mode on Windows OS.

Description:

Application performs activities requiring the user to have Administrator privileges in order to perform activities such as modifying the registry.

Solution:

Either remove activities requiring Admin permissions or check for Admin mode rights in a host configuration runtime phase executable and react accordingly. Application should not start if appropriate user permissions are not available.

An option is to use a package such as Thinstall to wrap the application in a stand alone executable containing the appropriate .NET resources.

14. IF #12: Application written in Java

Description:

Application requires presence of an appropriate JVM.

Solution:

There is no unified architectural solution for hosting a jvm on the U3 device at this time. It is the application's responsibility to do one or both of the following:

- Provide a standalone jvm with the application package.
- During host configuration, check to see if a suitable jvm is already present on the host machine. If it is not, inform the user of the issue including how to install the appropriate resources on the host machine.

15. IF #13: Requires .NET framework runtime package

Description:

Application is a .NET application requiring the .NET framework runtime package to be present.

Solution:

There is no unified architectural solution for hosting the .NET framework runtime package on the U3 device at this time.

During host configuration, check to see if the .NET framework runtime package is already present on the host machine. If it is not, inform the user of the issue. It is appropriate to direct the user to the appropriate online resource where they can find the .NET installer.

An option is to use a package such as Thinstall to wrap the application in a stand alone executable containing the appropriate .NET resources.

Return to Overview

Return to Overview

When to Use the DAPI

U3 Knowledge Base Article, ID# 118

Please refer to the article in the online Knowledge Base for any updates.

You do not need the DAPI to create a U3 smart application.

For most applications, the Application Deployment Guide is the sole must-read document inside of the SDK, as it will cover all you need to create most U3 smart applications. This type of application is termed U3LP in the Developer Guide.

U3 smart applications are managed by the U3 Launchpad which means that:

- The U3 Launchpad controls the application's lifecycle and provides it with a runtime environment
- At each stage of the application's lifecycle the developer can customize the application behavior by providing executables that get triggered automatically by the U3 Launchpad
- The U3 Launchpad handles the detection and processing of device connect and disconnect events for you, so you can focus on developing your application

Only use the DAPI if one or more of the following features are essential to your application:

- Device Query Functions
- Cookie functions
- U3 smart drive discovery

U3A (U3 aware) applications are host based applications that use the DAPI to discover U3 smart drives that are connected to the host.

U3LP+ applications are U3 smart applications that need some of the features of the DAPI described above.

The DAPI is needed in the following situations:

	Unique Device ID needed	**Cookies needed**	**U3 smart drive discovery**
U3LP			
U3LP+	Use DAPI	Use DAPI	Use DAPI
U3A	Use DAPI	Use DAPI	Use DAPI

When would an application need to use the Device Query Functions? This may apply, for instance, if your application has security needs or a licensing system that require access to the U3 smart drive's unique ID. (Section 4 of the DAPI Reference Guide)

When would an application need to use the Cookie Functions? If it is essential for you to have access to the Cookie area on the U3 smart drive, you will need to use the DAPI. (Section 6 of the DAPI Reference Guide)

Return to Overview

Return to Overview

When to Use the DAPI

When would an application need to discover U3 smart drives?

- If you are creating a U3 aware application (see the Developer Guide, section 1.2 for a description).
- If you are creating a U3 smart application that needs to detect the arrival of other U3 smart drives on the system.

Return to Overview

Return to Overview

U3 Platform 1.0 SDK
Application Deployment Guide

Version 1.0
December, 2006
Revision 6.0

Return to Overview

Return to Overview

U3 Platform 1.0 SDK
Application Deployment Guide

Table of Contents

Return to Overview

U3 Platform 1.0 SDK
Application Deployment Guide

U3 Platform 1.0 SDK
Application Deployment Guide

List of Figures

U3 Platform 1.0 SDK
Application Deployment Guide

List of Tables

Return to Overview

U3 Platform 1.0 SDK
Application Deployment Guide

1. Introduction

This document describes the full specifications for creating version 1.0 U3 smart applications and adapting Windows applications to run on the U3 platform Launchpad. Developers may additionally use the Developer Guide and/or DAPI Reference Guide. However, their usage is not a requirement for an application to be U3 compliant.

2. Document Scope

This document consists of 9 sections. The reader will first be introduced to U3 platform concepts and design considerations. The U3 Package definition will then be described in detail, followed by an in-depth specification of the U3 application lifecycle and its interaction with the U3 platform. Throughout this document, a worked example will be used to demonstrate the use and implications of the U3 platform requirements. A brief description of each section is provided below.

- Section 3, U3 Platform Overview, offers a brief introduction to the U3 platform, including a description of the U3 terminology used. This section highlights the difference between standard Windows and U3 smart applications, and describes the U3 Launchpad, the main console for U3 users.
- Section 4, U3 Application Overview, introduces the reader to the U3 application lifecycle and lists recommendations that are important to consider when developing a mobile application for the U3 platform.
- Section 5, Worked Example, introduces a sample application that is used throughout this document as a means of illustrating how an existing Windows application can be adapted to become U3 compliant.
- Section 6, U3 Package Specification, defines the structure and contents of the directories that incorporate the application installation package. The U3 Package defines how U3 applications are delivered and prepared for installation on a U3 device.
- Section 7, Manifest File Specification, defines the structure of a manifest file. The U3 Package specification defines a manifest file, which describes the U3 application contained in the U3 Package. The manifest file also provides information required by the U3 Launchpad to execute the application.
- Section 8, U3 Actions, defines U3 actions. U3 actions, defined in the manifest file, are possible events that may occur/be performed by the U3 Launchpad at each stage of the U3 application lifecycle.
- Section 9, U3 Runtime Variables, defines runtime variables for U3 applications. The U3 actions, combined with the U3 runtime variables, provide the framework for mobilizing applications. The runtime variables allow the U3 application to operate despite the dynamic nature of the operating environment, for example different operating systems, drive letters, and languages.

Return to Overview

- Section 10, Action Logging, describes how to get and interpret the debug information generated by U3 Launchpad version 1.2.

The following topics are beyond the scope of this document:

- **U3 Device API**: DAPI (Device API) can optionally be used by U3 applications to get low level device information, and perform device related operations. These topics are fully described in the *Developer Guide* and the *DAPI Reference Guide*.
- **U3 smart Application Smart Logo Compliance**: in order to ensure that software applications can be deployed on U3 smart drives, U3 LLC has developed the ***U3 smart Application Compliance Program***. This includes a set of test criteria, which must be followed before a software application can be certified and before the right to display the U3 smart logo can be granted. The kit is available from http://www.u3.com/developers/.
- **U3 smart Application Certification Self Test Criteria**: This document describes the set of test criteria that U3 Launchpad Application developers must pass before the application can be submitted for U3 smart logo compliance.

3. U3 Platform Overview

This section provides an overview of the U3 platform, including a description of the U3 terms and definitions used, and a comparison of the significant differences between standard Windows applications and U3 applications. This section also includes an introduction to the U3 Launchpad, the main console for U3.

3.1. Terminology

Table 1 lists the U3 terms and definitions used in this document.

Table 1: U3 Terms and Definitions

Term	Definitions
U3 smart device	A USB flash drive (UFD) that implements features according to the U3 device specification. Throughout this document, the U3 smart device will be referred to as a U3 device.
Host machine	The host computer (Windows PC or laptop) with inserted device(s) upon which U3 applications are executing.
U3 Launchpad	The main console application with which the user interacts. The U3 Launchpad runs automatically when a U3 device is inserted into the host machine. All user applications installed on the U3 device are accessed via the U3 Launchpad. The U3 Launchpad enables users to manage and configure U3 devices and U3 applications.
U3 application	An application that travels with the user on a U3 device. U3 applications run on all U3-compliant devices from all manufacturers. These applications are also known as U3 Launchpad (U3LP) applications or U3 smart Applications. The Device API (DAPI) will evolve so that applications for one generation of U3 devices will also run on future generations of U3 devices.

Term	Definitions
U3 Package	An application packed for installation on a U3 device. A U3 Package has a .u3p extension.
Manifest file	An XML file with a .u3i extension that describes the application files in a U3 Package, and details installation and execution instructions to be used by the U3 Launchpad.
U3 environment	The set of directories on the U3 device and temporary directories on the host machine that the U3 Launchpad maintains to execute and manage U3 applications.
U3 Download Central	The website maintained by U3 LLC that contains a searchable catalog of U3 applications.
Device API (DAPI)	Set of APIs that enables U3 applications to use the enhanced features of the U3-compatible device, receive events, and perform actions such as eject. DAPI is specified in the *DAPI Reference Guide*. An overview of DAPI can be found in the *SDK Developer Guide* and are included in the U3 SDK.

3.2. U3 Applications vs. Standard Applications

The difference between a U3 application and a standard Windows application is that a U3 application is mobile. There are three issues affecting mobile applications:

- The location of files and directories change from machine to machine. For example, the drive letter assigned to the U3 device may be [E:] on one machine and [F:] on another machine. The user name may be different, or the user's home directory may reside on a different local hard drive or even on a network drive.
- The host operating system may vary from machine to machine. Even two machines with the same operating system may have different versions of core libraries and services, etc.
- The device may be ejected while it is still being used by applications that are running.
- To enable mobility, U3 creates a runtime environment for each U3 application. The runtime environment is a set of directories and runtime variables managed by the U3 Launchpad that shields the U3 application from the dynamic nature of directory and file locations. The U3 application is therefore independent of drive letters and paths assigned by the host machine to the U3 device.
- The biggest conceptual difference between regular Windows applications and U3 applications is the installation process. With a standard Windows application, the installation process includes extracting the files, configuring the host machine, and updating the DLLs or services that are needed to satisfy the application's requirements on the host machine. After the installation is complete, the host machine is ready to execute the application.

- With U3 applications, the installation process is very different because the U3 application does not know in which operating environment it will run. The installation process of a U3 application is therefore split into two distinct phases:
    - The device installation phase, in which all U3 application files are extracted from the U3 package file to the U3 device.
    - The host configuration phase, which executes every time the U3 device is inserted into the host machine. In this phase, the host machine is configured as required by the U3 application. For example, if certain registry keys need to exist or certain DLLs need to be available, then these tasks are performed at the host configuration stage before the U3 application is executed.
- An important consideration to bear in mind is that each new host machine may require different actions to be performed during the host configuration phase. This may be due to variation between operating systems, or the existence of other applications on the host machine that have already provided the required resources.
- Before the U3 application is first run, designated files can be copied to the host machine by the U3 Launchpad to ensure that even if the U3 device is ejected while the U3 application is still running, the U3 application can close in an orderly manner.

3.3. The U3 Launchpad Application

The U3 Launchpad is the main console application with which the user interacts. The U3 Launchpad runs automatically when a U3 device is inserted into the host machine. All end user U3 applications installed on the U3 device are accessed through the U3 Launchpad.

The U3 Launchpad also enables users to manage and configure the U3 device and the U3 applications. The U3 Launchpad can be configured to run certain U3 applications automatically when the U3 device is inserted into a host machine.

Figure 1 illustrates the U3 Launchpad v1.0 and V1.2 interfaces.

U3 Platform 1.0 SDK
Application Deployment Guide

U3 Launchpad v1.0

U3 Launchpad v.1.2

Figure 1: The U3 Launchpad GUI

When the U3 Launchpad starts, it creates its U3 environment. Within this environment, runtime environments are created for the installed U3 applications. As new U3 applications are installed, new runtime environments are created on the host machine for those applications.

When the U3 device is ejected using the U3 Launchpad or simply by removing the U3 device from the host machine, the U3 Launchpad cleans up the host machine by removing the U3 environment and restoring the host machine to the state it was in before the U3 device was inserted. During the cleanup process, the U3 Launchpad requires applications to undo any system changes made by the U3 applications.

U3 actions, defined in the U3 application's manifest file (See the Manifest File Specification section), instruct the U3 Launchpad which commands to execute at each stage of the U3 application's lifecycle.

3.3.1. U3 Launchpad Version 1.2 versus 1.0

U3 released to smart drive manufacturers version 1.2 of the U3 Launchpad in July. The new version should replace U3 Launchpad v1.0 in their products in the 2006 Q4. In addition, users can upgrade their U3 Launchpad online.

Version 1.2 is 100% backward compatible with version 1.0 so your existing U3 Launchpad applications do not need to be modified to run on the new version. However, there are a couple of new features that you may want to take advantage.

From an application perspective, U3 Launchpad v1.2 adds the following features

- Application tooltips: there's a new property in the manifest file – shortDescription – which allows you to specify a tooltip the U3 Launchpad v1.2 displays when the user passes over your application. See the Manifest File Specification section for the details.
- Chinese support: We have added a Chinese support for the U3 application's metadata. See Appendix B for the LCID number.
- New Smart Drive Documents folders: To help users keep their files organized. See the User Data vs Application files description.
- Action Logging: U3 Launchpad can output debug information to a file or debugging tool. See the Action Logging section in this manual.

There are two other U3 Launchpad improvements that affect application developers indirectly:

- Improved U3 Software Central: The U3 smart application download site is easier to use. The web interface window now mimics standard browser conventions so users navigate the site just like they would with IE or Firefox.
- Improved U3 Launchpad user interface: U3 Launchpad v1.2 is easier to use with fewer menus, application setting menus available with right click, personalized drive labels, and, on some versions from some vendors, password protection.

This means your applications are easier to download and install from U3 Software Central and easier to manage and run.

4. U3 Application Overview

A U3 application is a Windows application that is designed or adapted to run from a U3 device through the U3 Launchpad. U3 compliant applications are referred to as U3 smart applications.

U3 Platform 1.0 SDK
Application Deployment Guide

U3 applications are available to users via U3 Download Central. Users use the U3 Launchpad to access U3 Download Central, which offers easy navigation and browsing of U3 applications that are ready to be downloaded to a U3 device. Software updates are also available via U3 Download Central.

This section introduces the U3 application lifecycle, which lays the foundation for understanding the methodology that must be implemented when building a mobile application. The next section provides a worked example that illustrates how to adapt a Windows application to run in the U3 environment.

Note: It is recommended that the reader review this section a second time after becoming familiar with the details that are described in the following sections.

4.1. U3 Application Lifecycle

The lifecycle of a U3 application consists of six distinct phases:

- Installing the U3 package on a U3 device
- Configuring the host machine
- Starting the U3 application
- Stopping the U3 application
- Cleaning up the host machine
- Uninstalling the U3 package from the U3 device

U3 applications can be installed from a U3 package file on the host machine, or directly from U3 Download Central to a U3 device. During the installation process, application files are extracted from the U3 package and placed on the U3 device in directories assigned for that U3 application. The set of directories, along with the application-specific runtime variables, are collectively known as the U3 application's runtime environment.

The host configuration stage allows the U3 applications to configure the host machine before they are run for the first time. Every time a U3 device is inserted into a new host machine, the host configuration action is executed before a U3 application runs. The host configuration stage can also test for minimal requirements needed by the U3 application to run. For example, user rights, OS service pack, etc. If these requirements are not met, the host configuration stage can request that the U3 Launchpad not run the U3 application.

Following successful host configuration, the U3 application can be started and stopped multiple times. As long as the U3 device stays in the host machine, the U3 application can be started and stopped as often as is required.

When a user chooses to eject the U3 device, the U3 Launchpad begins the cleanup process. Each U3 application that executed the host configuration stage now executes the host cleanup stage. The host cleanup stage allows the U3 application to undo any system changes it performed while it was running or during the host configuration stage. This may include undoing registry changes, removing files that were copied, and restoring general settings to their state prior to the U3 application's host configuration.

The U3 Launchpad then cleans up the U3 environment that it created for the U3 application on the host machine.

The final stage in a U3 application's lifecycle is uninstalling the U3 application from the U3 device. This includes removing the U3 application's package files and the runtime environment directories from the U3 device.

Figure 2 below illustrates an application lifecycle.

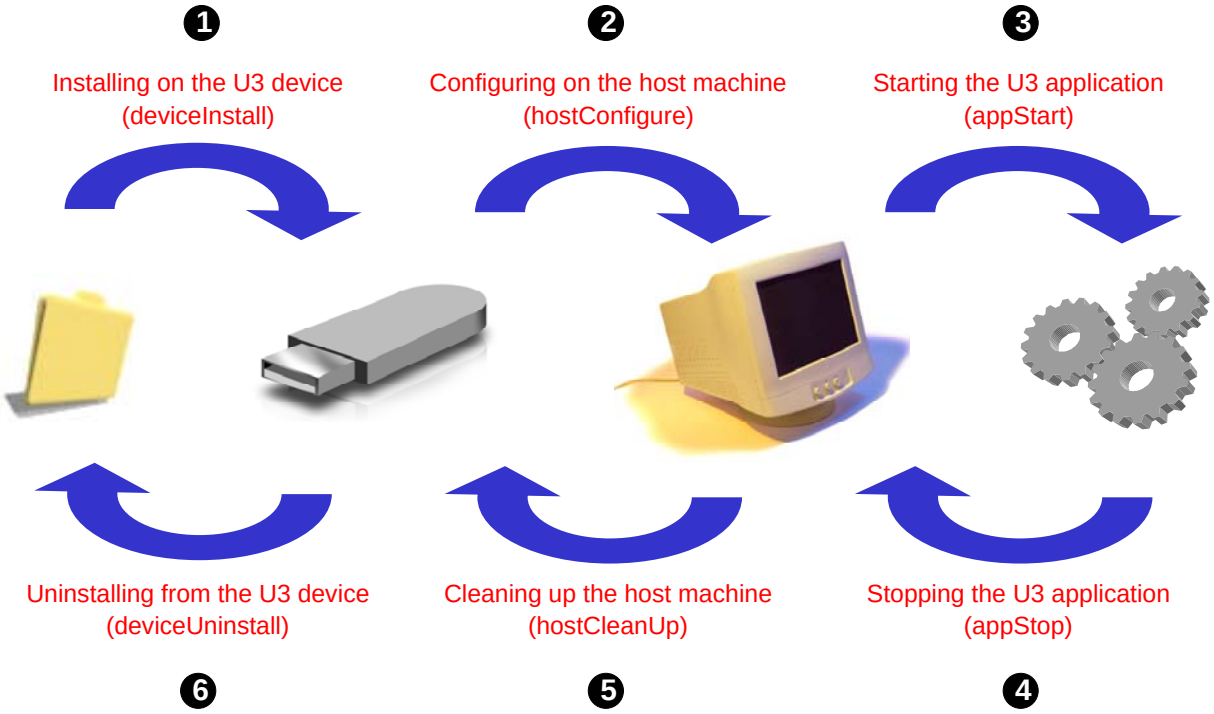


Figure 2: U3 Application Lifecycle

This deployment guide details the changes that have to be implemented for a U3 application to conform to the lifecycle. Each phase is handled by an action, and each action is described in detail in the U3 Actions section.

4.2. Supported Operating System

The U3 Launchpad supports the following host machine operating systems:

- Windows 2000, Service Pack 4 and later
- Windows XP, all versions and service packs
- Windows Server 2003, all versions and service packs

4.3. U3 Application Requirements

- U3 applications must be in the form of an executable (EXE) file. All DLL-based applications must be wrapped in an executable.
- DLLs and libraries required by U3 applications that are not part of the minimum installation of the supported operating systems must be packaged with the U3 application files.
- U3 does not protect one U3 application from editing a second U3 application's configuration data. The U3 application is responsible for testing the consistency and correctness of its configuration data.
- U3 applications must not perform actions that require the user to reboot the host machine or log out.

4.4. U3 Application Recommendations

U3 applications are mobile in nature. Some of the recommendations below may help developers improve the mobility of their applications in the U3 environment, and maintain application isolation when multiple U3 devices are inserted into a host machine.

- Where possible, paths in configuration files should be relative to the U3_* paths defined in Section 9.2. Path Variables.
- Files should be stored in the recommended directories defined in Section 6, U3 Package Specification.
- When explicitly loading a DLL or using any other files, to maintain application isolation, absolute paths should be calculated to ensure that the U3 application is loading the expected file.
- U3 applications should minimize the amount of time that file handles are held open on the U3_device.
- Use of the Windows registry should be avoided where possible. If the Windows registry is used, the U3 application should undo any changes before closing.
- Developers should assume that the U3 application will be executed from a different temporary directory/drive every time it is run.
- Developers should use a deterministic approach to handle cases in which the same U3 application is executed from different locations simultaneously (suc as two U3 devices inserted into the same host machine). This should take into account cleanup of artifacts and whether multiple instances can execute simultaneously. Strategies such as including a value unique to the U3 device in the registry path will ensure that applications running simultaneously from two devices will not interfere with the other application's registry data.
- Avoid using the keywords "install," setup," or update," in the filename of any of your U3 smart application's executables. When Vista thinks that an

application is an installation program, that process gets launched with administrator permissions. If this happens to one of your U3 manifest actions, and it runs with elevated permissions, the elevated process will not inherit the current user's U3 environment variables and this inevitably has a significant impact on your U3 smart application's execution. Consult the U3 Knowledge Base for updated information on this issue.

5. Worked Example

The worked example in this section describes how to adapt a fictitious existing application so it becomes a U3 application. This example is provided to help developers that are familiar with Windows application design to use the U3 design methodology. This example shows alternate file location strategies in addition to showing options for supporting and adapting registry usage.

5.1. Application Description

CWmaster is a Windows-based crossword game. It is already up to version 3. CWmaster (CWM) is freeware and has a large user community. Users can create new crosswords, which can then be uploaded to the community website (http://www.cwmaster.org/). CWM can download new puzzles from the site. Users must have accounts on the website to download and upload new crosswords.

CWM uses the registry to store server details, account details, and the list of downloaded crosswords. Downloaded crosswords are stored in a subdirectory underneath the user's application data directory. Each user on the machine has his own set of puzzles.

For each user, CWM stores the amount of time taken by the user to complete the puzzle, completeness score, and other statistics. This data is stored in the registry.

CWM is C++ based and does not use any COM. The application consists of a main executable and 4 application-related DLLs. It also uses MFC with dynamically linked DLLs. CWM runs on all Windows operating systems.

The developer has access to all the source code and has been tasked with adapting CWM to make it U3 compatible. There is a requirement that the U3 version of CWM not interfere with or use information belonging to the host version of the application that is already installed on the host machine.

Currently only one instance of CWM is allowed to run at a time. When CWM loads, it checks to see if other instances are running. If CWM detects another instance, it shuts down with no warning message. This is still a requirement for the U3 application, whether the other instance of CWM has started from the same U3 device, another U3 device, or the host machine. If a –stop parameter is provided to cwm.exe, it will stop the existing running instance.

CWM uses the following registry locations:

HKCU\Software\CWM\
HKCU\Software\CWM\Server
HKCU\Software\CWM\Account
HKCU\Software\CWM\Puzzles
HKCU\Software\CWM\Score

The crosswords are downloaded to the following Windows directory:

%AppData%\CWMaster\puzzles\

Puzzles that the user has created are stored by default in the following location:

%My Documents%\My Crosswords\

The application is installed to %ProgramFiles%\CWMaster\ (typically C:\Program Files\CWMaster\). Within this directory are the application files cwm.exe, xmdb.dll, board.dll, sync.dll, and cw.dll. The files tutorial.avi and cwm.hlp are also stored in this directory.

5.2. Migration Analysis

This section elaborates on the U3 application requirements and recommendations. Implementation considerations are described in detail to demonstrate architectural thinking behind U3 applications.

- DLLs and libraries required by applications that are not part of the minimum installation of the supported operating systems must be packaged with the U3 application files.

  The U3 application relies on the MFC DLLs, as it is dynamically linked. The installer of the Windows version of CWM copies these files to the host at install time if they do not exist. A mobile version would require that these be installed every time the U3 device is inserted into a new host machine. Implementation options are as follows:

  - Copy the files to the system32 directory the first time CWM is run on a new host machine. This may be problematic as U3 requires that the application remove all system changes when it stops running, and the files may be in use by another application. In addition, copying files to the system32 directory may require administrator rights.
  - Place the DLLs in the same directory as CWM.
  - Statically link the MFC.
- U3 does not protect one application from editing a second application's configuration data. Each U3 application is responsible for testing the consistency and correctness of its configuration data.
  CWM currently uses the unique directories and registry entries created by Windows for the **current user** to ensure that data files for each user are kept separately.

There may be multiple U3 devices inserted to the host machine at the same time with CWM installed. Therefore, the logged-in user account that is now common to all inserted U3 devices is no longer a valid method for separating two different sets of user and application data.

Care must be taken to ensure that if any data (either file or registry based) is written by CWM to the host machine, identification information should be included to ensure that the data is not accidentally used by another instance of CWM.

Implementation options are as follows:

- When writing data to files or the registry on the host machine, add a device identifier into paths, e.g. \[device id]\
- Do not write data to the host machine, but store all data on the U3 device

- U3 applications must not perform actions that require the user to reboot the host machine or log out.

  Not applicable.

- Where possible, paths in configuration files should be relative to the U3_* paths defined in Section 7.5, Applying the Worked Example.

  The paths to the current set of directories used by the non-U3 version of CWM is obtained either from the registry or via Windows environment variables. The U3 application will use the U3 variables in their place to determine the location of each U3 directory that it will use to read and write files.

- Files should be stored in the recommended directories.

  As CWM is now mobile, any data that must remain with the U3 application when the U3 device is moved from one host machine to another should be stored on the U3 device.

  CWM currently recommends that users store saved crosswords that they are creating underneath the My Documents\My Crosswords directory. The U3 practice is for the application to prompt the user to store these files on the U3 device, ideally in a \My Crosswords directory on the U3 device Documents directory.

- When explicitly loading a DLL or using any other files to maintain application isolation, absolute paths should be calculated to ensure that the U3 application is loading the expected file.

  CWM should ensure that it loaded the DLLs that sit with the executable by including the full path in the DLL name.

- Applications should minimize the amount of time that file handles are held open, particularly those of files on the U3 device.

This should not be an issue as CWM does not hold any files open. Puzzle files are read into memory and then closed. Edited puzzles are stored in memory until they are saved.

- Use of the Windows registry should be avoided where possible. If this registry is used, the U3 application should undo any changes before closing.

  CWM relies heavily on the registry. Nearly all the data that is written to the registry must travel with the U3 application on the U3 device. Implementation options are as follows:

  - Move all registry entries to a U3 device-based file and update the U3 application implementation.
  - Continue to use the registry with minimal code changes in the main U3 application. The registry should be restored from the U3 device to the host machine before CWM is executed, and copied back to the U3 device when CWM stops. The data can also be written to the U3 device periodically to ensure that the U3 device copy is updated even if the U3 device is removed before CWM can close and update the file in an orderly manner. As stated in requirement 2, if the registry is used, an additional element must be inserted into the registry key path to ensure that it is unique per U3 device, in addition to being unique to the current Windows user.

- The developer should assume that the U3 application will be executed from a different temporary directory/drive every time it is run.

  CWM should use the U3 variables to ensure that it does not use any hard-coded paths to the U3 device in its configuration files.

6. U3 Package Specification

A U3 Package is an archive file containing all the information and files required to install a U3 application on a U3 device, and run the U3 application from the U3 device.

The U3 Package is a compressed zip file that uses a .u3p extension in place of the .zip extension.

The U3 application package is installed on the U3 device using the U3 Launchpad; the U3 Launchpad can install a U3 Package from a local drive, or download the package from U3 Download Central and install the U3 application.

The U3 Package file is comprised of the following four sections.

- Manifest
- Data
- Host

- Device

Each section refers to a directory in the root of the U3 Package file. Files must be placed in the correct directory according to their role. Directories may contain sub-directories. Other than the Manifest directory, the remaining directories may be empty. Empty directories do not need to be included in the U3 application package.

Each directory helps the U3 application maintain its mobility and stability. Some directories are on the U3 device, and others are temporary directories on the host machine.

Every U3 application has a unique identifier that is the same for all versions of that application. Unique IDs help the U3 Launchpad correctly identify the U3 application regardless of its name. Unique IDs are used in upgrades, licensing, and so on. If a new version of the U3 application has a different Unique ID, it is considered by the U3 Launchpad to be a different application, regardless of whether or not it has the same name as a previous version. The Unique ID is specified in the manifest file (see Appendix A, Generating a unique id).

The root of a U3 Package may only contain the directories listed in Table 2.

Table 2: Permitted U3 Package Directory Labels

Descriptive Directory Name	Package Directory Label	Description	Extracted To	U3 Reference Variable
Manifest	manifest	Package configuration files	n/a	n/a
Application data	data	Application persistent configuration files	The device	U3_APP_DATA_PATH
Host exec	host	Application executable and related files	The host machine	U3_HOST_EXEC_PATH
Device exec	device	Application auxiliary files	The device	U3_DEVICE_EXEC_PATH

Any files outside of these directories are ignored.

6.1. Manifest Directory

The manifest directory contains all the configuration files for the U3 Package. A first-generation U3 (version 1.0) application package requires that two files be placed in the manifest directory; the manifest file (manifest.u3i) and the application icon (*.ico).The manifest file describes the U3 application's properties and actions to perform at each stage of the application's lifecycle. The icon file is used by the U3 Launchpad in the program list it displays.

Table 3 lists the manifest files that should be included in the manifest directory for the first generation of U3 Packages.

Table 3: Manifest Files

File Name	Definitions
manifest.u3i	An XML manifest file describing application attributes, install options and actions.
*.ico	The application icon to use in the U3 Launchpad program list. The name of this file must be listed in the manifest file.

See the Manifest File specification section for the full description of the manifest file.

6.2. Application Data Directory

The application data directory contains all the U3 application's persistent configuration files. This directory may contain all configuration files associated with the U3 application that require persistent changes through the use of the application. For example, skins, template files, license files, playlists, user preferences, and any file that should remain with the U3 application when it is upgraded.

The files are extracted from the U3 Package data directory when the U3 application is downloaded. They are then installed on the U3 device in the application data directory designated for the U3 application. Additional files can be added to the application data directory on the U3 device at runtime.

As the application data directory is device-based, it helps U3 applications maintain mobility, because storing the configuration data in this directory enables moving from host to host with the U3 application. The path of the application data directory is calculated by the U3 Launchpad using the U3 application's Unique ID, and is kept in a runtime variable named U3_APP_DATA_PATH.

The application data directory supports update rules that define how files in the application data directory should be treated during a U3 application update. The update rules are defined in the manifest file.

Lifespan of the application data directory:

- Created when the U3 application is installed on the U3 device.
- Removed or overwritten when the U3 application is upgraded, according to the update rules in the manifest file.
- Deleted when the U3 application is uninstalled from the U3 device.

Note: Runtime configuration files that are specific to an individual host machine should not be stored in the application data directory. It is recommended that these files be stored in the U3_HOST_EXEC_PATH directory.

Table 4 demonstrates the state of the contents of the application data directory on the U3 device during the U3 application lifecycle.

Table 4: Device Data Directory Lifecycle

Application Data Directory	Package Install	Application Configuration	Cleanup	Application Uninstall	Application Upgrade
Content State	Extracted	---	---	Deleted	Update rules

6.3. Host Exec Directory

The host exec directory contains the U3 application executable and related files (such as DLLs) that are required to execute it. Before the U3 application is executed, the U3 Launchpad extracts the U3 application files from the U3 Package host directory to a temporary directory on the host machine, called the host exec directory. Any DLL or additional files that the U3 application needs at runtime must be included in this directory, or optionally in the device exec directory. Only core DLLs that ship with the supported operating systems need not be included with the U3 Package.

By copying the executable files to the host exec directory, U3 application stability can be maintained even if the U3 device has been removed from the host machine. All the files required to maintain the stability of the U3 application remain available.

The path of the host exec directory on the host machine is calculated using the U3 device serial number and the U3 application's Unique ID. This path is kept in a runtime variable called U3_HOST_EXEC_PATH.

Sub-directories may exist underneath the host exec directory in the U3 Package file. The path, relative to the host directory, is preserved when the files are copied to the temporary directory on the host machine.

Lifespan of the host exec directory:

- Created when the U3 application is executed.
- Deleted when the U3 device is removed from the computer.

All files in the host exec directory should be considered volatile, meaning any changes to this directory may not be preserved for the next time the U3 application executes, as they are extracted again to the host exec directory each time the U3 application is run. If the U3 application requires that certain files maintain persistent changes (for example, configuration files) then this data should be placed in either the U3 Package's device exec or application data directory.

Table 5 demonstrates the state of the contents of the host exec directory on the host machine during the U3 application lifecycle.

Table 5: Host Execution Directory Lifecycle

Host Exec Directory	Package Install	Application Configuration	Cleanup	Application Uninstall	Application Upgrade
Content State	Extracted	Extracted	Deleted	Deleted	Deleted and re-extracted

6.4. Device Exec Directory

The device exec directory contains the U3 application auxiliary (non-configuration) files that the U3 application stores on the U3 device for use at runtime. The device exec directory stores static files used by the U3 application that do not have to be copied to the host machine to maintain the stability of the U3 application. For example, help files, media files, libraries and other infrequently used files, and large files that do not have to be fully copied to the host machine each time, like game level files or dictionaries. The U3 application can copy these files as needed. Files that may be built at install time but are not configuration files can also be stored in the device exec directory.

When the U3 application package is installed on the U3 device, the files in the device directory are extracted from the U3 application package to the device exec directory on the U3 device. Changes to files and additions to the device exec directory are persistent and travel with the U3 application from host to host.

The path to the device exec directory is calculated using the U3 application's Unique ID and is kept in a runtime variable called U3_DEVICE_EXEC_PATH.

The device exec directory supports update rules, which define how files in the directory should be treated during an application update. The update rules are defined in the manifest file.

Lifespan of the device exec directory:

- Created when the U3 application is installed.
- Removed or overwritten when the U3 application is upgraded, according to the update rules in the manifest file.
- Deleted when the U3 application is uninstalled and upgraded.

Table 6 demonstrates the state of the contents of the device exec directory on the U3 device during the U3 application lifecycle.

Table 6: Device Execution Directory Lifecycle

Device Exec Directory	Package Install	Application Configuration	Cleanup	Application Uninstall	Application Upgrade
Content State	Extracted	---	---	Deleted	Update rules

6.5. Application Directories Lifecycle

When mobilizing an application, care must be taken when considering the placement of each of the application files in the package directories. The Removable Disk partition on the U3 device is configured with two root directories:

- Documents for user files
- System for application files

Within the System directory is an Apps directory. Each application gets a separate directory entry in the Apps directory with Data and Exec folders.

Table 7 summarizes the data lifespan for each application directory.

Table 7: Application Directories Lifecycle

	Package Install	Application Configuration	Cleanup	Application Uninstall	Application Upgrade
Device Data Directory	Extracted			Deleted	Update rules
Device Exec Directory	Extracted			Deleted	Update rules
Host Exec Directory	Extracted	Extracted	Deleted	Deleted	Deleted and re-extracted

6.6. User Data Files vs. Application Files

The three directories discussed in Table 7 are provided to store U3 application files, that is, the files needed by U3 applications to execute. Any files stored in these directories are deleted when the U3 application is uninstalled. U3 applications should never prompt the user to save data files in any of the application directories.

Data files created by the user during application use, such as backups, documents, music, and pictures should be stored in the Documents directory. If the Smart Drive is running Launchpad 1.0, there are no subfolders created for the Documents folder.

If the Smart Drive is running Launchpad 1.2 the Documents folder has the following subfolders

- Pictures
- Downloads
- Video
- Music

U3 does not define where user data files should be stored. Ideally, these files should be stored in a subdirectory underneath the Documents directory on the U3 device relative to the U3_DEVICE_DOCUMENT_PATH. U3 applications should start at this user data directory when opening the File Open or File Save (As) windows.

U3 applications can create subdirectories in the Documents directory on the U3 device. This is typically done at install time.

6.7. Applying the Worked Example

section, CWM contains the following files and registry entries:

C:\Program Files\CWMaster\cwm.exe

C:\Program Files\CWMaster\xmdb.dll

C:\Program Files\CWMaster\board.dll

C:\Program Files\CWMaster\sync.dll

C:\Program Files\CWMaster\cw.dll

C:\Program Files\CWMaster\tutorial.avi

C:\Program Files\CWMaster\cwm.hlp

C:\Windows\System32\mfc70.dll

C:\Documents and Settings\{user}\Application Data\CWMaster\puzzles\1024.xml

C:\Documents and Settings\{user}\Application Data\CWMaster\puzzles\1218.xml

C:\Documents and Settings\{user}\Application Data\CWMaster\puzzles\2127.xml

C:\Documents and Settings\{user}\My Documents\My Puzzles\my1puzzle.xml

HKCU\Software\CWM\version=3.0.1

HKCU\Software\CWM\Server\primary=http://www.cwmaster.org/checknew.php

HKCU\Software\CWM\Server\secondary=http://www.cwmaster1.org/checknew.php

HKCU\Software\CWM\Account\userid=12122292

HKCU\Software\CWM\Account\hash=02A7FG53A2232ED1D3C

HKCU\Software\CWM\Puzzles\basedir=%MyDocuments%\My Puzzles\

HKCU\Software\CWM\Puzzles\1024\filename=1024.xml

HKCU\Software\CWM\Puzzles\1024\timestamp=20050107.125621

HKCU\Software\CWM\Puzzles\1024\difficulty=4

HKCU\Software\CWM\Puzzles\1329\filename=1218.xml

HKCU\Software\CWM\Puzzles\1329\timestamp=20050204.230111

HKCU\Software\CWM\Puzzles\1329\difficulty=3

HKCU\Software\CWM\Puzzles\2127\filename=2127.xml

HKCU\Software\CWM\Puzzles\2127\timestamp=20050205.075640

HKCU\Software\CWM\Puzzles\2127\difficulty=4

HKCU\Software\CWM\Score\overall=4,24,1216,16

HKCU\Software\CWM\Score\Puzzels\1024=7,29,1542,18,20050122

HKCU\Software\CWM\Score\Puzzels\1218=7,21,1423,21,20050206

HKCU\Software\CWM\Score\Puzzels\2127=4,25,1209,16,20050206

6.7.1. Handling File Locations

The first step in handling file locations is identifying files that are required in the host exec directory on the host machine to ensure U3 application stability. The following files have been identified as having to be placed in the host exec directory: cwm.exe, xmdb.dll, board.dll, sync.dll, and cw.dll. The mfc70.dll file should also be placed with the U3 application files in the host exec directory, as it is required to avoid changing the system32 directory.

The help and tutorial files may be in the same directory as the files listed above; however, there is no need to copy them to the host machine each time the application is started. These files do not affect stability and are used only infrequently. Therefore the best place for them is the device exec directory on the U3 device.

The puzzle files that are downloaded should be stored on the U3 device to ensure that they travel with the U3 application. As the files are only opened temporarily and then immediately closed, they do not have to sit in the host exec directory as they do not affect stability. They can therefore be stored in either the application data or device exec directory. The puzzle files can also be placed in a new directory in the Documents directory of the U3 device. If the puzzle files are stored in the application data or device exec directory, they might get deleted at upgrade time due to the update rules. If the puzzle files are stored in a new directory, they will remain untouched when the U3 application is updated or uninstalled. CWM is freeware, and therefore has no obligation for backward compatibility. The XML format of the puzzles has changed over time, resulting in old puzzles not being readable by newer versions of CWM. Taking all these arguments into consideration, it has been decided to store the downloaded puzzles in a subdirectory underneath the device exec directory.

That is, C:\Documents and Settings\{user}\Application Data\CWMaster\puzzles\ will be replaced with %U3_DEVICE_EXEC_PATH %\puzzles.

Note: If CWM is uninstalled from the U3 device, the downloaded puzzles are also removed.

The puzzles created by the user may be stored within the CWM application directories. This means that the puzzles are deleted with CWM at uninstall time. However, the standard for puzzles has been adopted by other communities and new tools now exist for editing these files. The user data files should therefore be stored in a common location so that all applications can edit them.

The files will be stored in a subdirectory of the device called \My Puzzles, i.e.:
C:\Documents and Settings\{user}\My Documents\My Puzzles\ will be replaced with %U3_DEVICE_DOCUMENT_PATH%\My Puzzles\

6.7.2. Handling Registry Entries

CWM uses the Windows registry extensively. There are two possible methods of handling registry use when mobilizing an application.

One approach is to stop using the Windows registry and to move to a file-based configuration store, located in the application data directory. A data structure may be used that matches the registry layout. CWM only opens the file for reading and writing; if at any time the file is not available, CWM can use default values to recover gracefully.

An alternative approach is to keep the main copy of the settings on the U3 device. When the host machine is configured, these values can typically be copied to the registry. During the host cleanup phase, the updated registry values are copied back to the U3 device. However, this approach is problematic in two ways:

- The U3 device may be removed from the host machine at any time. The solution is to ensure that any important registry changes are immediately written back to the U3 device. Alternatively, the registry can be synchronized with the U3 device at regular intervals.
- Application collision may occur. It is possible that a host version of CWM and another instance of CWM on one or more U3 devices may all read and write to the same registry keys. This can occur regardless of whether the application instances execute at the same time or at different times. This scenario can cause application instability or result in data being transferred between application instances. The solution is to add a unique value, often based on a hardware device attribute such as device serial number, to the key path. Each application instance therefore has a unique set of registry keys. This solution also allows for registry keys to be cleaned up or

removed with the knowledge that no other instances of CWM rely on these keys.

Both approaches for handling registry entries store the persistent copy of the configuration data in a file in the CWM data directory on the U3 device. This file, called cwm.cnf, contains default values. When CWM is first run, the user is prompted for addition details, such as account information, difficulty level, etc.

6.7.3. Helper Applications

Since the lifecycle of the mobile CWM application is different from the Windows version, a new executable called install.exe must be created. This application is called when CWM is installed on the U3 device, and when CWM is uninstalled.

When run at install time, CWM prompts the user to confirm the location and name of the \My Puzzles directory. The selected directory name and path are written to the cwm.cnf file.

While possible, it was decided not to prompt the user for account details at install time, as the code already exists in the main application. The user can be prompted if CWM cannot find the user's account details.

At uninstall time, install.exe is called with an –uninstall parameter. CWM prompts the user to confirm application uninstall, and also allows the user to opt to delete the \My Puzzles directory created at install time.

The install.exe is placed in the device exec directory.

6.7.4. The CWmaster Lifecycle

Using the assessment detailed in previous sections, the preliminary layout of the CWM U3 Package is summarized in Table 8.

Table 8: Worked Example File Placement

File Name	Files Contained
Manifest	manifest.u3i ,cwm.ico
Host	cwm.exe, xmdb.dll, board.dll, sync.dll, cw.dll, mfc70.dll
Device	tutorial.avi, cwm.hlp, install.exe
Data	cwn.cnf

The six general stages of a U3 application lifecycle are described in Section 4.1, U3 Application Lifecycle. This section tracks the application files throughout the CWM lifecycle, while noting the U3 Launchpad activity at each stage.

Installing the U3 Package on a U3 Device

Table 9 describes the process of installing a U3 application (CWM) to a U3 device. First, the U3 Launchpad extracts the files from the U3 Package to each of the CWM's runtime environment directories. After the U3 Launchpad has extracted the files, the U3 Launchpad runs the U3 device install action. The device install action is defined in the manifest file as executing the install.exe application. Install.exe creates the \My Puzzles directory or equivalent, according to user choice.

Table 9: Installing CWM on a U3 Device

	Application Data	Device Exec	Host Exec	<Device>\ Documents\ My Puzzles
U3 Launchpad Activity	Create directory. Extract files from package	Create directory. Extract files from package	Create directory. Extract files from package	
Action (install.exe)				Create directory
File snapshot	cwm.cnf	tutorial.avi, cwn.hlp, install.exe	cwm.exe, xmdb.dll, board.dll, sync.dll, cw.dll, mfc70.dll	

Configuring the Host Machine

The host configuration stage is not necessary if the registry is replaced by a file. However, if CWM still uses the registry, this step is used to populate it based on the information in the cwm.cnf file in the application data directory.

As the first step in the host configuration phase, the U3 Launchpad creates the host exec directory and extracts the files from the U3 Package. Following this, the action defined to execute for hostConfigure is always executed (see the hostConfigure action description for more information).

Table 10 describes the scenario of a U3 device being inserted into a host machine.

Note: This scenario describes inserting a U3 device after CWM has already been installed on the U3 device.

Table 10: Configuring the Host Machine with CWM Data

	Application Data	Device Exec	Host Exec	<Device>\ Documents\ My Puzzles
U3 Launchpad Activity	n/a	n/a	Create directory. Extract files from package.	
Action (none)	None			
File	cwm.cnf	tutorial.avi, cwn.hlp, install.exe	cwm.exe, xmdb.dll, board.dll, sync.dll, cw.dll, mfc70.dll	

Starting the U3 Application

CWM is now running. When CWM downloads new puzzles, they are stored in the device exec directory under \Puzzles. In Table 11, note that CWM has downloaded puzzles 1024.xml, 1218.xml, and 2127.xml. The user has also created his own puzzles, which were saved on the U3 device in the directory that was created by install.exe at install time.

Cwm.cnf is edited at runtime, and changes are kept in the application data directory on the U3 device.

Table 11: Starting CWmaster

	Application Data	Device Exec	Host Exec	<Device>\ Documents\ My Puzzles
U3 Launchpad Activity	No activity			
Action (none)	User configuration data is updated in the configuration file.	Downloaded puzzles are stored by CWM in this directory.		Puzzles created by the user are stored here.
Files	cwm.cnf	tutorial.avi, cwn.hlp, puzzles/1024.xml, puzzles/1218.xml, puzzles/2127.xml	cwm.exe, xmdb.dll, board.dll, sync.dll, cw.dll, mfc70.dll	My1puzzle.xml, sundaycw.xml

Stopping the Application

In the first three stages of the U3 application lifecycle, the U3 Launchpad performs a task and then calls the action as defined in the manifest file. In the last three stages, the action is executed first and then the U3 Launchpad performs a follow-up task.

The appStop action is responsible for stopping the running the application. The U3 Launchpad does nothing at this stage besides executing the stop command. The stop command may copy back the registry to the U3 device after CWM has stopped.

Table 12: Stopping CWmaster

	Application Data	Device Exec	Host Exec	<Device>\ Documents\ My Puzzles
Action (cwm.exe -stop)				
U3 Launchpad Activity	No activity			
Files	cwm.cnf	tutorial.avi, cwn.hlp, puzzles/1024.xml, puzzles/1218.xml, puzzles/2127.xml	cwm.exe, xmdb.dll, board.dll, sync.dll, cw.dll, mfc70.dll	My1puzzle.xml, sundaycw.xml

Cleaning Up the Host Machine

If there are registry keys, an application must be created to remove them. No additional cleanup is required by CWM, as it did not write any files to the host machine outside of the host exec directory.

After the host cleanup is complete, the U3 Launchpad deletes the host exec directory.

Table 13: Host Cleanup Stage

	Application Data	Device Exec	Host Exec	<Device>\ Documents\ My Puzzles
Action	None			
U3 Launchpad Activity			Delete directory	
Files	cwm.cnf	tutorial.avi, cwn.hlp, puzzles/1024.xml, puzzles/1218.xml , puzzles/2127.xml		My1puzzle.xml, sundaycw.xml

Uninstalling the U3 Package from a U3 Device

CWM may prompt to delete the "My Puzzles" directory that was created at install time. After the action has finished, the U3 Launchpad deletes the U3 application's runtime environment directories.

Table 14: Uninstalling the U3 Package

	Application Data	Device Exec	Host Exec	<Device>\ Documents\ My Puzzles
Action (install.exe -uninstall)				May delete directory according to user's choice.
U3 Launchpad Activity	Delete directory	Delete directory	Delete directory	
Files				My1puzzle.xml, sundaycw.xml

6.8. Creating the U3 Package File

This section describes the process of creating a U3 Package file. The procedure requires the following steps:

- Analyze the software file structure and define which files should be placed in each of the designated directories.
- Update the software to refer correctly to the relative file locations for the install, run, and uninstall procedures (the U3 environment variables, defined

in Section 9, U3 Runtime Variables, help to dynamically determine these locations).

- Create the manifest directory, and optionally create the data, host exec and device exec directories, described in the previous sections. Ensure that the name and case of each directory exactly corresponds to the name definition of each directory.
- Place the corresponding files into each directory. The manifest directory **MUST** contain the manifest file and an icon file. The other directories may be empty.
- Select the four directories and create a zip file. Empty directories do not have to be included in the U3 Package. The zip file should have all the selected directories in the root directory.

  Note: Zip compression must be used. Other formats, such as .arj and .rar, will not be recognized by the U3 Launchpad.

- Rename the output ZIP file by replacing the .zip suffix with a .u3p suffix

7. Manifest File Specification

The manifest.u3i file describes the U3 application properties (e.g. name, vendor), installation instructions, and action commands. The manifest file contents are defined using XML tags.

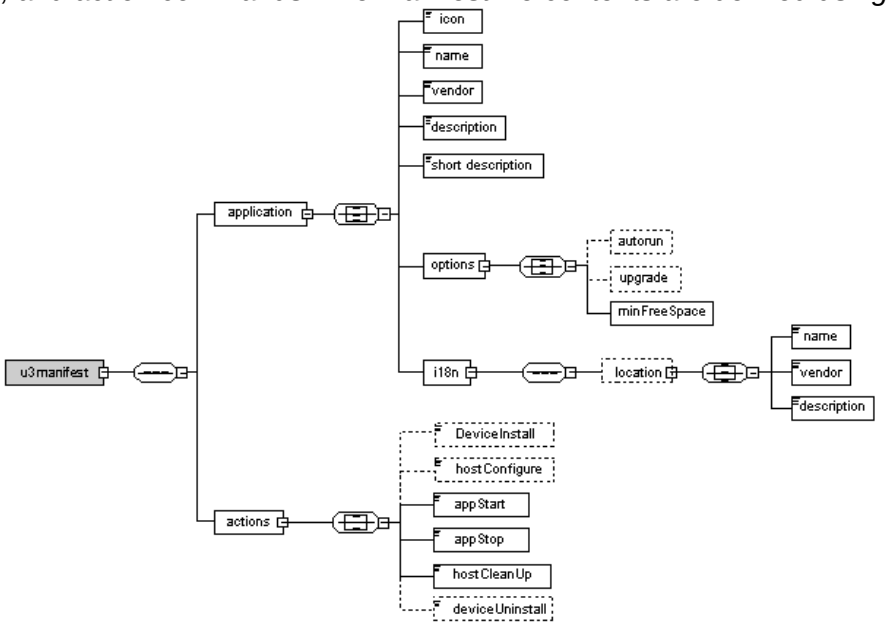


Figure 3 illustrates a scheme of the nested structure.

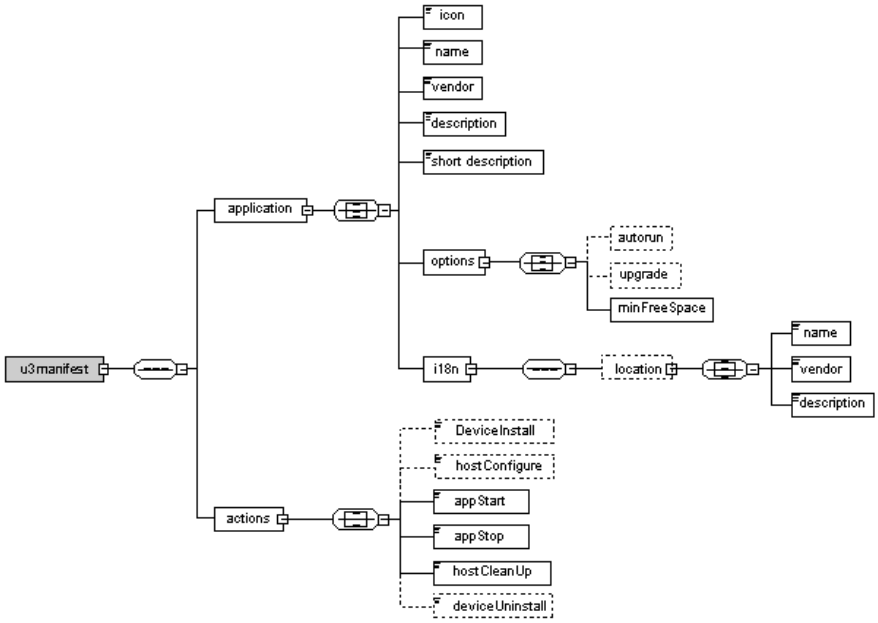


Figure 3: Manifest File Structure

Note: Elements with dotted outlines are optional.

Each tag in the nested structure in Table 7 is described in the sections that follow. The tag values and attributes are also detailed and illustrated. Sample code of the relevant tag is shown at the end of each section.

7.1. u3manifest

The u3manifest tag defines the start of a U3 manifest file. The manifest file contains two sections:

- Application: Description of the U3 application, defining the name, Unique ID, version, and so on.
- Actions: Commands to execute at each stage of the lifecycle.

Attributes

Version	The only valid value is “1.0”. Identifies the manifest as a first-generation U3 application.

Sample Code

```
<u3manifest version="1.0">
<application>...</application>
<actions>...</actions>
</u3manifest>
```

7.2. u3manifest\application

This section describes the U3 application. The application tag contains attributes that uniquely identify the U3 application. All tags within the application tag are used by the U3 Launchpad when it displays U3 application information to the user. In addition, U3 Download Central uses the information in the Application Catalog. An English description of the U3 application is a minimal requirement; however, U3 application descriptions can be provided in other languages. If the current system language matches a non-English U3 application description, it will be used instead of the default English description.

Attributes

uuid	Required	A unique 128-bit value that identifies the application. All versions of the same application must use the same uuid. The format is a "registry format GUID," for example, A402EAB1-ADAE-4018-BFCF-BC4D9D9A9F83. The uuid is generated and defined by the application developer. See Appendix A, Generating a unique id for instructions on generating the value.
version	Required	The current version of the application. The format is "X.X.X.X", for example, 1.13.0.21. Only integers 0-9 are acceptable. The U3 Launchpad uses this value to determine if there is a new version available and if the version being installed is newer, older, or the same as the current version.

Sample Code

```
<application uuid="A58038B1-02C0-4ce5-AFCF-2CD8F2FEA357"
version="0.0.0.1">
        <icon>…</icon>
<name>…</name>
<vendor>…</vendor>
<description>…</description>
<options>…</options>
<i18n>…</i18n>
</application>
```

7.2.1. u3manifest\application\icon

The icon tag defines the name of the icon file in the manifest directory of the U3 Package. This tag should include the relative path from the manifest directory. The U3 Launchpad uses the icon file in the displayed applications list.

The format of the icon file should be 32x32 pixels in 256 colors. The U3 Launchpad can adapt and resize the icon if it is not in the correct format. The file name must have a .ico extension.

Sample Code

<icon>myapp.ico</icon><icon>myapp.ico</icon>

7.2.2. u3manifest\application\name

The name tag defines the English name of the U3 application. Names longer than 24 characters may be truncated by the U3 Launchpad.

Sample Code

<name>A Sample Application</name>

7.2.3. u3manifest\application\vendor

The vendor tag defines the English name of the U3 application vendor. The vendor tag has a 24 character limit.

Attributes

url	optional	If defined, a link is created from the vendor name to the defined URL. When the user clicks the vendor name in the U3 Launchpad, a browser window opens pointing to the specified URL.

Sample Code

<vendor url="http://simpleappcorp.com">Sample Apps are Us</vendor>

7.2.4. u3manifest\application\description

The description tag defines the English description of the U3 application. The description is displayed in U3 Download Central, on the first page of the installation wizard, and in the Manage Programs window. Only text is supported, and HTML formatting elements are ignored. Descriptions longer than 200 characters may be truncated.

Sample Code

<description>A very simple description</description>

7.2.5. u3manifest\application\shortDescription

The shortDescription tag defines a tooltip for the application. The tooltip is displayed when the user drags the pointer over the application in the Main Window. Only text is supported, and HTML formatting elements are ignored. Descriptions longer than 200 characters may be truncated.

Note: If no shortDescription is defined, the tooltip is taken from the description tag.

Sample Code

<shortDescription>A very simple description</shortDescription>

7.2.6. u3manifest\application\options

The options tag defines the install options that the U3 Launchpad checks and uses during U3 application installation and upgrade. The following options are supported:

- Upgrade
- Autorun
- MinFreeSpace

7.2.6.1. u3manifest\application\options\upgrade

The upgrade tag defines whether this version can upgrade a previous version of the U3 application. If the upgrade tag is not defined, then an existing application version cannot be upgraded and must be removed before this version can be installed.

The appData and deviceExec attributes define how the files in the application data directory and device exec directory are processed during an upgrade.

Table 15: appData and deviceExec Attributes

Attribute Value	Description
remove	The application data/device exec directory is removed before the new U3 application files are extracted from the U3 Package.
overwrite	All the application data/device exec files in the U3 Package overwrite any existing files in the application data/device exec directory.
add	Files that already exist in the application data/device exec directory are not extracted from the U3 Package. Files that do not exist in the application data/device exec directory are extracted and added to the existing files.

Both attributes must be specified in the upgrade tag.

Sample Code

```
<upgrade appData="overwrite" deviceExec="add"/>
```

7.2.6.2. u3manifest\application\options\autorun

The autorun tag defines whether or not the U3 application starts automatically when the U3 device is inserted in the host machine. If this tag is defined, then the autorun option is enabled for the U3 application. The user can enable and disable autorun for any U3 application in the Manage Programs window in the U3 Launchpad. If the autorun tag is not defined, the default value in the installation wizard is “do not autorun.”

Note: Defining this tag does not guarantee that a U3 application will autorun upon U3 device insertion. It only recommends to the user to enable autorun when prompted to do so in the installation wizard.

Sample Code

```
<autorun/>
```

7.2.6.3. u3manifest\application\options\minFreeSpace

The minFreeSpace tag defines the free space required on the U3 device to successfully install the U3 application. The size of the free space is specified in megabytes (MB), and may be a decimal representation (for example, 1.2, 7, 3.5). The calculation of the area should include the size of the U3 package file and files in the application data and device exec directories after the deviceInstall action has completed. Ideally, the size of the recommended free space should be rounded up to allow for different sector sizes on the U3 device.

Note: When the U3 Launchpad installs a U3 application, it may require more than minFreeSpace space on the U3 device to allow installation. If that space is not available, the installation may fail. For example, if a U3 application defines minFreeSpace=15, the U3 Launchpad may require a minimum of 17MB free space on the U3 device due to administrative actions that the U3 Launchpad performs when it installs an application. So if the U3 device has only 16MB free, although sufficient for storing the U3 application itself, the U3 Launchpad may refuse to install the U3 application and the installation process may fail.

Sample Code

```
<minFreeSpace>3.5</minFreeSpace>
```

7.2.7. u3manifest\application\i18n

The i18n (internationalization) tag allows the U3 application's metadata to be defined in additional languages. The i18n tag may contain multiple location tags, each specifying the location ID (LCID) it supports. See Appendix B, Supported International Location IDs.

Sample Code

```
<i18n>
      <location lcid="1036">
        …
      </location>
      <location lcid="3082">
        …
      </location>
</i18n>
```

7.2.7.1. u3manifest\application\i18n\location

The location tag defines the location ID for the name, vendor, and description tags defined within.

Attributes

lcid	Required	The decimal LCID of the included text. The LCID defines both the region and language. For a list of LCIDs, see Appendix B, Supported International Location IDs.

Sample Code

```
<location lcid="3082">
   <name> la mejor fabricación del software </name>
   <vendor url="http://simpleappcorp.com/es/">La muestra Apps es
nosotros</vendor>
   <description>Heche una ojeada este uso fresco</description>
</location>
```

7.2.7.1.1. u3manifest\application\i18n\location\name

The name tag defines the application name in the language defined in the lcid attribute of the parent location tag.

Names over 24 characters in length may be truncated by the U3 Launchpad.

7.2.7.1.2. u3manifest\application\i18n\location\vendor

The vendor tag defines the vendor name in the language defined in the lcid attribute of the parent location tag. Names over 24 characters in length may be truncated by the U3 Launchpad.

Attributes

url	Optional	If defined, a link is created from the vendor name to the defined URL. When the user clicks on the vendor name in the U3 Launchpad, a browser window opens pointing to the specified URL. If this URL is not defined and a URL in the topmost vendor tag is defined, then the URL in the topmost vendor tag is used.

7.2.7.1.3. u3manifest\application\i18n\location\description

The description tag defines the application description in the language defined in the lcid attribute of the parent location tag. Names over 24 characters in length may be truncated by the U3 Launchpad.

7.3. u3manifest\actions

The actions tag defines the U3 actions that the U3 Launchpad should execute at each of the following stages of the U3 application lifecycle:

- device install
- host configure
- application start
- application stop
- host cleanup
- device uninstall

7.3.1. Action Format

The format of each action is identical:

<actionName workingdir=”path to working directory” cmd=”path to executable”>

command line options

</actionName>

Each action must define the path to the executable that the U3 Launchpad must execute.

If the working directory attribute is defined, then the working directory is set to this value, otherwise the U3_HOST_EXEC_PATH directory is used.

The command parameter should only include the full path to the executable. The U3 Launchpad assumes that only the executable name is specified and not any command line parameters. When the U3 Launchpad creates the action command, it surrounds the action with quotation marks. If there are any command line parameters in the command value, this creates an illegal command line string that the U3 Launchpad is unable to execute. All command line parameters should be placed in the body of the action element.

Sample Code

WRONG

<appStart cmd="%U3_HOST_EXEC_PATH%\main.exe –silent"></appStart>

This code causes the U3 Launchpad to try and execute the application `"%U3_HOST_EXEC_PATH%\main.exe –silent".`

CORRECT

<appStart cmd="%U3_HOST_EXEC_PATH%\main.exe">-silent</appStart>

This code causes the U3 Launchpad to execute the following: `"%U3_HOST_EXEC_PATH%\main.exe" –silent`

7.3.2. Environment Variables

In both the cmd and workingdir attributes, as well as the tag body, U3 and Windows environment variables can be used. All strings surrounded by % symbols are interpreted as environment variables, and are processed before the command is executed.

Variable names are first compared with known U3 variables. If a match is not found, the variable names are then compared with Windows environment variables. If no match is found, the variable is replaced with an empty string. If the variable name is defined as both a U3 environment variable and a Windows environment variable, the U3 variable will always be used.

7.3.3. Escaping Characters

If the command requires XML tag characters, such as input stream "<", output stream ">", and so on, the command may be surrounded by a CDATA command.

Sample Code

```
<action cmd=”%U3_HOST_EXEC_PATH%\myapp.exe”
><![CDATA[> “%TEMP%\log.txt”]]></action>
```

Note: In the parameter definitions (for example, > “%TEMP%\log.txt”) all path variables should be enclosed with quotation marks (“) to ensure that spaces in the path are correctly interpreted.

7.3.4. Supported Actions

Table 16 lists the supported action names.

Table 16: Supported U3 Actions

Action	Type	Description
deviceInstall	Optional	Executes this command when the application is installed on the U3 device. See Section 8.2.1., deviceInstall.
HostConfigure	Optional	Executes this command before the U3 application is executed for the first time after the U3 device has been inserted into the host machine. See Section 8.2.2., hostConfigure.
appStart	Required	Executes this command to start the core U3 application. See Section 8.2.3., appStart.
appStop	Required	Executes this command to stop running U3 application. See Section 8.2.4., appStop.
hostCleanUp	Optional	Executes this command to clean up any changes made to the system by hostConfigure and the U3 application. See Section 8.2.5., hostCleanUp.
Device Uninstall	Optional	Executes this command to clean up any related files on the U3 device before the U3 application is removed. See Section 8.2.6.,deviceUninstall.

7.3.5. Action Code Examples

Consider the following scenario:

My simple application has a run_me.exe file that starts the application. Run_me.exe receives parameters that are predefined in the cmd.txt file that was in the data directory in the application package, and was extracted to the application data directory on the device. Parameters in this file are separated using the % character.

A friendly user has just installed my simple application package on his U3 device and is now choosing to run the application. The user is on a public computer, and is logged in as 'kiosk' and the U3 device drive letter is [D:].

The following is the U3 environment:

The U3_HOST_EXEC_PATH (where my simple application execution files were extracted to by the U3 Launchpad) is: C:\Documents and Settings\kiosk\ApplicationData\U3\12A34B56C78D900E\A402EAB1-ADAE-4018-BFCF-BC4D9D9A9F83\Exec

The U3_APP_DATA_PATH (where my simple application configuration files are stored) is D:\System\Apps\A402EAB1-ADAE-4018-BFCF-BC4D9D9A9F83\Data

Example 1

Assume the deviceInstall action reads as follows:

<deviceInstall cmd="%U3_HOST_EXEC_PATH%\install.exe"> cmdfile="%U3_APP_DATA_PATH%\cmd.txt" sepchar=% cmd1=%NAME1%</deviceInstall>

The working directory is set to the default %U3_HOST_EXEC_PATH%, which maps to `C:\Documents and Settings\kiosk\ApplicationData\U3\12A34B56C78D900E\A402EAB1-ADAE-4018-BFCF-BC4D9D9A9F83\Exec` and the following command line is executed:

```
“C:\Documents and Settings\kiosk\ApplicationData\U3\12A34B56C78D900E\A402EAB1-ADAE-4018-BFCF-BC4D9D9A9F83\Exec\install.exe” cmdfile=”D:\System\Apps\A402EAB1-ADAE-4018-BFCF-BC4D9D9A9F83\Data\cmd.txt” sepchar=% cmd1=
```

Note: There is no value provided to cmd1 as %NAME1% did not map to a valid U3 or Windows environment variable.

Example 2

Assume the appStart action reads as follows:

```
<appStart
cmd="%ProgramFiles%\Internet Explorer\iexplore.exe">start.htm</appStart>
```

Internet Explorer is executed and displays start.htm. The current working directory is U3_HOST_EXEC_PATH, and all file paths in start.htm will be relative to this directory.

7.4. Sample Manifest File

A sample of a fully defined manifest file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<u3manifest version="1.0">
  <application uuid="A58038B1-02C0-4ce5-AFCF-2CD8F2FEA357" version="1.2.0.1">
    <icon>myApp.ico</icon>
    <name>A Sample Application</name>
    <vendor url="http://simpleappcorp.com">Sample Apps are Us</vendor>
    <description>A very simple description</description>
    <options>
      <autorun/>
      <upgrade appData="overwrite" deviceExec="add"/>
      <minFreeSpace>3.5</minFreeSpace>
    </options>
    <i18n>
      <location lcid="3082">
        <!-- Spanish. This is an example of specifying just by the sublanguage, rather
             then the location. Alternate could be 3082 -->
        <name> la mejor fabricación del software </name>
        <vendor url="http://simpleappcorp.com/es/">La muestra Apps es nosotros</vendor>
        <description>Heche una ojeada este uso fresco</description>
      </location>
    </i18n>
  </application>
  <actions>
    <deviceInstall  cmd="%U3_HOST_EXEC_PATH%\install.exe" >
       cmdfile="%U3_APP_DATA_PATH\cmd.txt" sepchar=%</deviceInstall>
```

```
    <hostConfigure
cmd="%U3_HOST_EXEC_PATH%\conf_tool.exe">datadir="%U3_APP_DATA_PATH%"</hostConfig
ure>
    <appStop  cmd="%U3_HOST_EXEC_PATH%\main.exe"/>
    <appStart  cmd="%U3_HOST_EXEC_PATH%\main.exe">-start</appStart>
    <hostCleanUp  cmd="%U3_HOST_EXEC_PATH%\cleanup.exe">-
upgrade=%U3_IS_UPGRADE%</hostCleanUp>
    <deviceUninstall workingdir="%U3_APP_DATA_PATH%"
        cmd="%ProgramFiles%\Internet Explore\iexplorer">uninstall.htm</deviceUninstall>
  </actions>
</u3manifest>
```

7.5. Applying the Worked Example

In this section, creating a manifest file will be exercised. The worked example will be used.

When creating the manifest file, it is recommended that the relevant parameters be listed in a table. Organizing the parameters in the type of list described in Table 17 make it easy to then create the file. See Appendix C, Sample Parameter List for Manifest Files.

Table 17: Parameter List for the U3 Manifest File

Parameter	Description
Application Name	Crossword Master
Version	3.0.0.10
GUID	6C8DD541-489F-4223-9166-74A09910D635
Website	www.cwmaster.org
Vendor name	International Crossword Community (ICC)
Vendor URL	http://www.cwmaster.org/
Description	The world's most famous crossword puzzle. Create your own puzzles!
Additional descriptions	None. This tool is English only.
Default autorun	No
Upgrade	No, this is the first public U3 version. It should not upgrade any previous version.
Min free space	10MB
Device Install	Install.exe from the device exec path

Parameter	Description
Host configure	No need
Application start	Cwm.exe from the host exec path
Application stop	Cwm.exe from the host exec path with a –stop parameter
Host cleanup	No need
Device uninstall	Install.exe from the device exec path with a –uninstall parameter

```
<?xml version="1.0" encoding="UTF-8"?>
<u3manifest version="1.0">
  <application uuid="6C8DD541-489F-4223-9166-74A09910D635" version="3.0.0.10">
    <icon>cwm.ico</icon>
    <name>Crossword Master</name>
    <vendor url="http://www.cwmaster.org/">International Crossword Community (ICC)</vendor>
    <description>World's most famous crossword puzzle. Create your own puzzles!</description>
    <options>
      <minFreeSpace>10</minFreeSpace>
    </options>
    <i18n>
    </i18n>
  </application>
  <actions>
    <deviceInstall workingdir="%U3_DEVICE_EXEC_PATH%"
cmd="%U3_DEVICE_EXEC_PATH%\install.exe" />
    <appStart  cmd="%U3_HOST_EXEC_PATH%\cwm.exe"/>
    <appStop   cmd="%U3_HOST_EXEC_PATH%\cwm.exe">-stop</appStop>
    <deviceUninstall cmd="%U3_DEVICE_EXEC_PATH%\install.exe">-uninstall</deviceUninstall>
  </actions>
</u3manifest>
```

8. U3 Actions

The U3 Launchpad runtime actions enable U3 applications to run in the U3 environment. The U3 application's manifest file is used to define the command for each action.

Each action is executed as a separate process by the U3 Launchpad, using either the default working directory or the directory defined in the manifest file for that action.

The six actions are grouped into three pairs of matching actions:

- **deviceInstall/deviceUninstall**: Adding/removing the U3 application from the U3 device.
- **hostConfigure/hostCleanUp**: Setting up the host machine to run U3 applications and removing all traces afterwards.
- **appStart/appStop**: Starting and stopping the U3 application on the host machine.

See Section 7.3, u3manifest\actions for additional information about U3 actions.

Note: The appStop and hostCleanUp action implementations must be located in the host exec directory.

8.1. Variables

The runtime variables can be used as command line parameters if they are surrounded by % symbols.

For example, MyApp.exe –dir=%U3_HOST_EXEC_PATH%. The variables are also available at runtime as environment variables. When querying the environment variables, it is not necessary to include the % symbols in the variable name.

The U3 Launchpad attempts to interpret all variables surrounded by % symbols in the command line and working directory attributes. The U3 Launchpad first attempts to compare the value with the U3_* variables. If this fails, the U3 Launchpad then attempts to compare the value with the system environment variables. If no match is found, the value is replaced with an empty string. If a variable name is defined as both a U3 and a Windows environment value, the U3 definition is used.

Note: It is recommended to enclose all paths that may contain spaces with quotation marks (“).

8.2. Action Definitions

8.2.1. deviceInstall

Required	No
U3 Launchpad Action upon Return Value	Zero: No action Non-zero: Roll back installation

The deviceInstall action allows the U3 application to perform any one-time set of tasks associated with installing the U3 application on the U3 device. These tasks include configuring files in the U3_APP_DATA_PATH and U3_DEVICE_EXEC_PATH directories, creating a user data directory on the U3 device, generating a license

key, U3 application activation, obtaining user details and preferences, and so on.

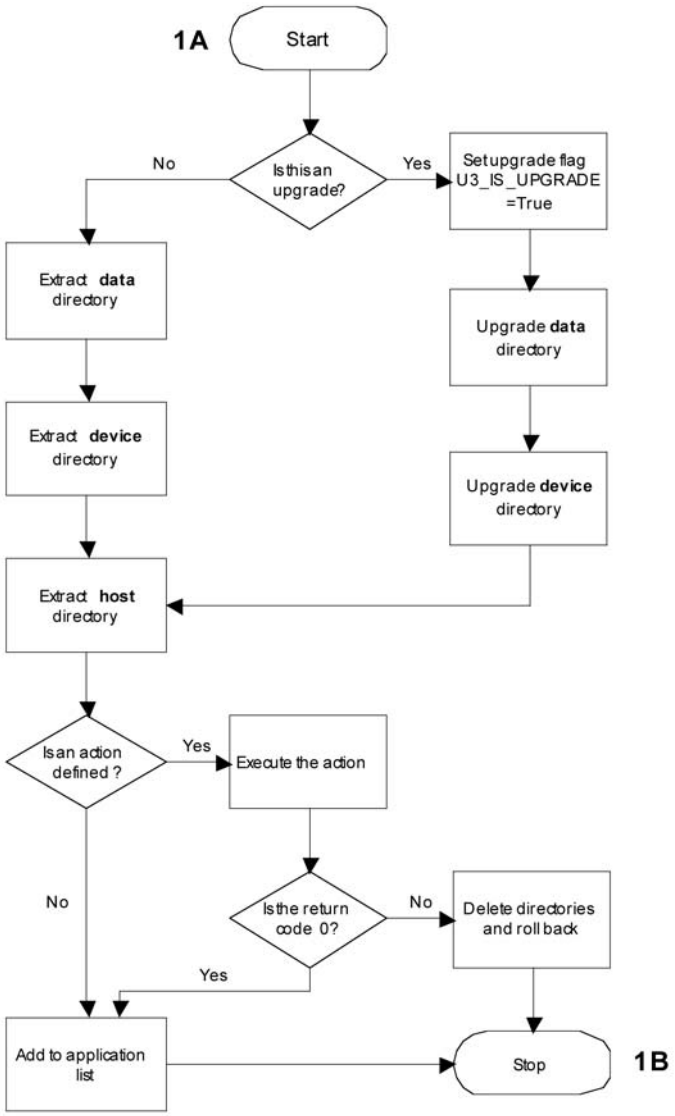


Figure 4: Installing the U3 Application (Stage 1)

Requirements

- If the deviceInstall action returns a non-zero result, the U3 Launchpad rolls back the installation of the U3 application package. **In this case, the deviceInstall action must clean up any temporary files and registry entries that it created**

or changed, as neither the deviceUninstall nor hostCleanUp actions are executed.

- The deviceInstall executable must execute without requiring the hostConfigure action to be executed.

Information

- The U3 Launchpad extracts the files from the U3 Package to the U3_APP_DATA_PATH and U3_DEVICE_EXEC_PATH directions on the U3 device, and U3 application files are extracted to the U3_HOST_EXEC_PATH temporary directory before deviceInstall is executed.
- The U3 Launchpad checks if the installed U3 application already exists on the U3 device, using the U3 application's Unique ID (UUID). If the U3 application already exists, then deviceInstall is part of an application upgrade and the UE_IS_UPGRADE flag is set to true by the U3 Launchpad. See the hostCleanUp and deviceUninstall.
- The U3 Launchpad extracts the files from the U3 Package to the designated directories before the actions are run. Therefore, when deviceInstall is run in an application upgrade, the files in the application data directory will have already been updated according to the update rule in the manifest file.

8.2.2. hostConfigure

Required	No
U3 Launchpad action upon Return Value	Application (appStart) is disabled if non-zero

The hostConfigure action is called every time a U3 application is extracted from a U3 device to a host machine. This action allows a machine-specific configuration to be performed before the U3 application is initially executed.

Typically, hostConfigure executes the first time the U3 application is run on the host machine. Before the U3 application can run for the first time (meaning the appStart action executes) the host machine must be configured. hostConfigure is only executed once before hostCleanUp is called. hostCleanUp is called when the U3 device is removed or a U3 application is uninstalled/upgraded.

hostConfigure can be used to test that the host machine provides the prerequisites required by the U3 application. hostConfigure may return a non-zero value, which disables the U3 application, If it determines that the U3 application cannot or should not run on the

host machine. For example, the operating system may not be supported, or the current user does not have the rights required by the U3 application.

Additionally, hostConfigure can prepare the host machine for the U3 application. This may include populating or updating registry entries, or copying files to the correct location.

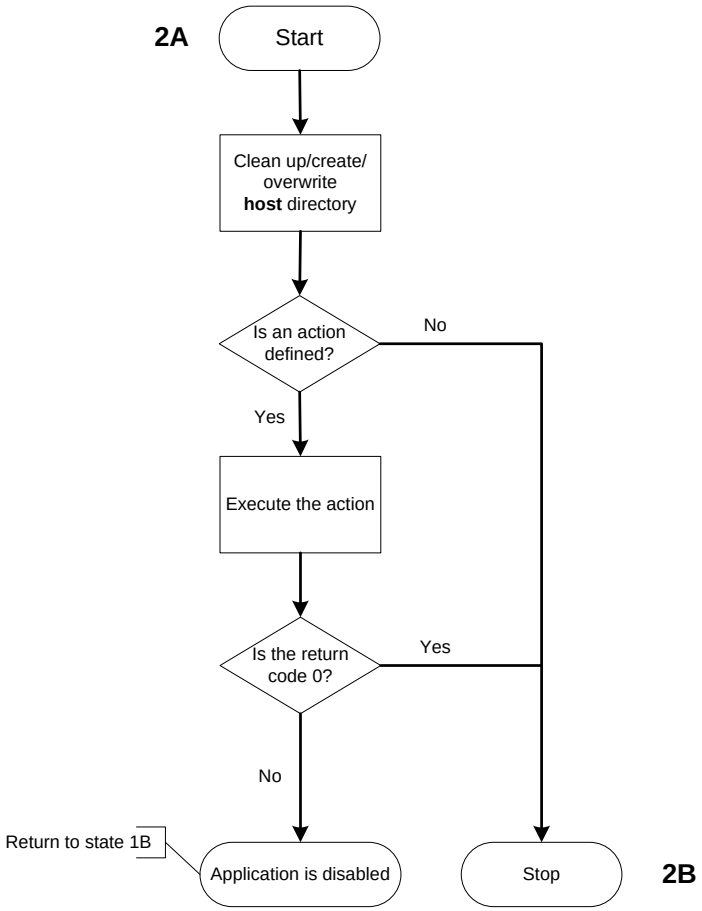


Figure 5: Configuring the Host Machine (Stage 2)

Requirements

- Any system changes performed during the hostConfigure action must be undone during the hostCleanUp action.
- The hostConfigure action should clean up the host machine if it intends to return a non-zero return value. The hostCleanUp action is not called if the hostConfigure action fails.

Information

- If hostConfigure returns a non-zero result, the U3 Launchpad will not call appStart. The U3 Launchpad notifies the user that the application failed to start. The user can attempt to launch the U3 application again via the U3 Launchpad, which will call hostConfigure again.
- Usage example: The U3 application may have downloaded extensions. These must be located in the host exec directory at runtime, but are stored on the U3 device in the device exec directory to ensure that they are persistent. hostConfigure can copy these files to the host exec location before the U3 application is executed.
- During hostConfigure, the U3 Launchpad extracts the host exec directory from the U3 Package to the host machine. hostConfigure overwrites existing files as a precaution.

8.2.3. appStart

Required	Yes
U3 Launchpad Action upon Return Value	N/A

The appStart action is used to execute the core U3 application. If any pre-configuration is required before the U3 application can execute, this should be implemented during the hostConfigure action.

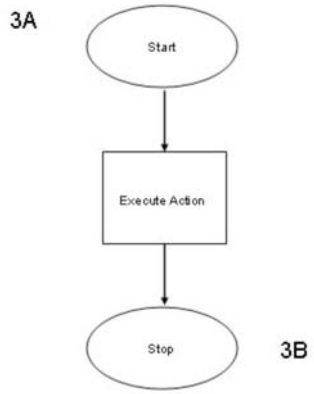


Figure 6: Starting the U3 Application (Stage 3)

Requirements

- appStart MUST return a zero return value result.

Information

- The U3 Launchpad runs each U3 application in a separate process space.
- The U3 Launchpad executes the defined command line and watches the process ID. The process ID is used by the U3 Launchpad to determine if the U3 application is still running.

8.2.4. appStop

Required	Yes
U3 Launchpad Action upon Return Value	N/A

The appStop action is used to help the U3 Launchpad shut down U3 applications in an orderly fashion. When a U3 application shuts down, the appStop action should clean up and close any open data files the user is working on, and remove any temporary files or registry settings from the host machine.

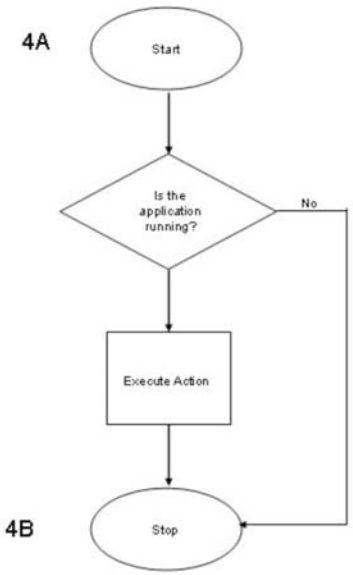


Figure 7: Stopping the U3 Application (Stage 4)

Requirements

- appStop MUST return a zero return value result.
- It is recommended that appStop does not return until it ensures that the U3 application has stopped. If appStop returns before the main application shuts down, the U3 Launchpad may launch hostCleanUp before the U3 application is closed.

- The appStop executable must be stored in the host exec directory to ensure it is available even if the U3 device is ejected.

Information

- If the U3 application creates a new process and closes the original process, then the U3 Launchpad is unable to determine that the U3 application is executing. The U3 Launchpad never calls appStop in this situation. However, hostCleanUp is always called, and the appStop logic should be moved to hostCleanUp if the U3 application does not maintain the process ID.
- If the U3 device is ejected before appStop is called, the U3_DEVICE_AVAILBLE flag is set to false.

8.2.5. hostCleanUp

Required	Yes
U3 Launchpad Action upon Return Value	N/A

The hostCleanUp action is called every time a U3 application's runtime environment is removed from a host machine. This happens at U3 application uninstall, U3 application upgrade, and U3 device removal. The hostCleanUp action should clean up any system and registry changes made by the U3 application to the host machine.

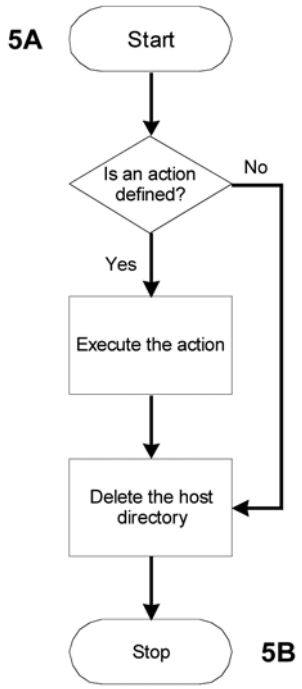


Figure 8: Cleaning up the Host Machine (Stage 5)

Requirements

- hostCleanUp MUST return a zero return value result.
- The hostCleanUp process should leave the host system in the same state as it was before the U3 application was extracted to the U3_HOST_EXEC_PATH location, prior to hostConfigure being executed.
- hostCleanUp may leave machine-specific configuration files only if they are located under the U3_HOST_EXEC_PATH directory. When the U3 Launchpad deletes this directory, there should be no more files associated with the U3 application on the host machine.
- hostCleanUp should ensure that it removes only those files and system settings that were created by the specific instance of the U3 application that was running, and not some other instance of the same application on the host machine.
- The hostCleanUp action implementation must be placed in the host exec directory.

Information

- If the U3 application does not maintain its process ID, then only hostCleanUp is called during shutdown and clean-up, and appStop is not called. U3 applications that do not maintain the process ID should implement appStop code in the hostCleanUp action.
- When a U3 application is being uninstalled, if hostConfigure was executed successfully (I.e., returned zero) for the U3 application, then hostCleanUp is called before deviceUninstall. In this case, the U3 Launchpad does not delete the host exec directory on the host machine until after deviceUninstall has been called.
- If hostConfigure was called during an upgrade, hostCleanup is called with the U3_IS_UPGRADE set to true. This notifies the U3 application to back up relevant files and perform any other required preparations for an application upgrade.
- Use the U3_* defined paths to maintain U3 application isolation.
- If the U3 device is ejected before hostCleanUp is called, the U3_IS_DEVICE_AVAILABLE flag is set to false.

8.2.6. deviceUninstall

Required	No
U3 Launchpad Action upon Return Value	Cancel uninstall if non-zero

The deviceUninstall action allows actions to be performed before the U3 application files are removed from the host machine and U3 device. The deviceUninstall action is used by the U3 Launchpad when a user chooses to uninstall a U3 application from his U3 device. The U3 Launchpad first executes the command that is defined in the manifest file to run when the deviceUninstall action is initiated. If the command is successful (returns a zero result), the deviceUninstall action removes the device exec and application data directories from the U3 device. If the command returns a non-zero result, then the U3 Package removal process is cancelled and no further action is taken.

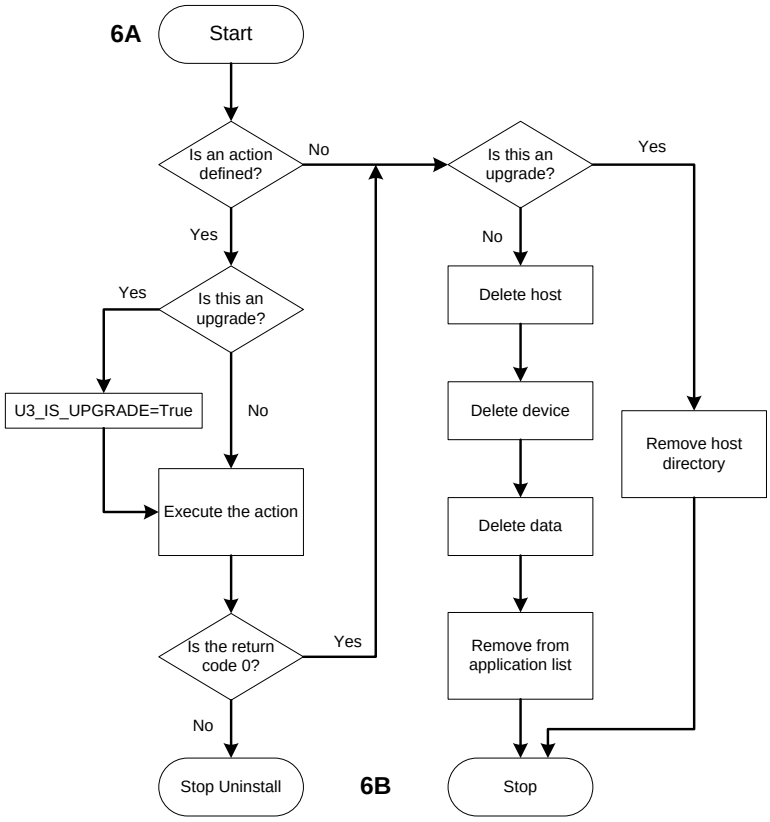


Figure 9: Uninstalling the U3 Application (Stage 6)

Requirements

- The deviceUninstall action should undo any changes the deviceInstall action made to the U3 device, in addition to any application-specific tasks, e.g. un-registering the application.
- If the Device API (DAPI) was used to write cookies, these should be removed during the deviceUninstall action.
- The deviceUninstall action should execute without requiring the hostConfigure action to be executed.

Information

- deviceUninstall may interact with the user. It may prompt the user to confirm removal of the U3 application.
- Before deviceUninstall is executed, appStop is executed if the U3 application is running.

- If hostConfigure has been executed, then hostCleanUp is executed before deviceUninstall is called.
- After deviceUninstall is executed, the U3_HOST_EXEC_PATH, U3_DEVICE_EXEC_PATH and U3_APP_DATA_PATH directories are removed.
- If a U3 application is being upgraded, deviceUninstall is called with the U3_IS_UPGRADE flag set to true. In this situation, the uninstall action typically does not perform any actions. However, deviceUninstall can be used to prepare the U3 application for upgrade, back up configuration and user data files, and so on.
- In an upgrade procedure, if deviceUninstall returns a non-zero error code, then the upgrade is cancelled.
- If the U3 application created files on the U3 device or a sub-directory in the device Documents directory, it can prompt the user to delete these or remind the user that files will be left behind in a specific location.

8.3. Applying the Worked Example

As described in Helper Applications, install.exe was created.

When run at install time, the action prompts the user to confirm the location and name of the \My Puzzles directory. The selected directory name and path is written to the cwm.cnf file.

At uninstall time, install.exe is called with a –uninstall parameter. The action prompts the user to confirm application uninstall, and also allow the user to delete the \My Puzzles directory created at install time.

If install.exe returns a non-zero result at install, the install process is cancelled. Similarly, if the install.exe –uninstall command returns a non-zero value, the uninstall action is cancelled.

Install.exe is placed in the device exec directory, and therefore the working directory is set to the device exec directory. The command is %U3_DEVICE_EXEC_PATH%\install.exe, but could have been simply install.exe as the working directory was already set.

```
<actions>
    <deviceInstall workingdir="%U3_DEVICE_EXEC_PATH%"
cmd="%U3_DEVICE_EXEC_PATH%\install.exe" />
    <appStart  cmd="%U3_HOST_EXEC_PATH%\cwm.exe"/>
    <appStop   cmd="%U3_HOST_EXEC_PATH%\cwm.exe">-stop</appStop>
```

```
    <deviceUninstall cmd="%U3_DEVICE_EXEC_PATH%\install.exe">-uninstall</deviceUninstall>
</actions>
```

At uninstall time, the install.exe –uninstall command is executed from the host exec directory. The U3 Launchpad checks for the return value. If it is zero, the U3 Launchpad removes CWM from the application list, and deletes all the directories belonging to CWM from both the host machine and the U3 device. If the return value is non-zero, the U3 Launchpad stops the uninstall procedure, and directories are not removed from either the host machine or the U3 device.

8.4. The Upgrade Process

This flowchart in Figure 10 complements the information regarding the upgrade process as documented above. There are three U3 actions that are potentially involved in the upgrade process as well as optional manifest.u3i file entries that control upgrade actions during the "deviceInstall" action.

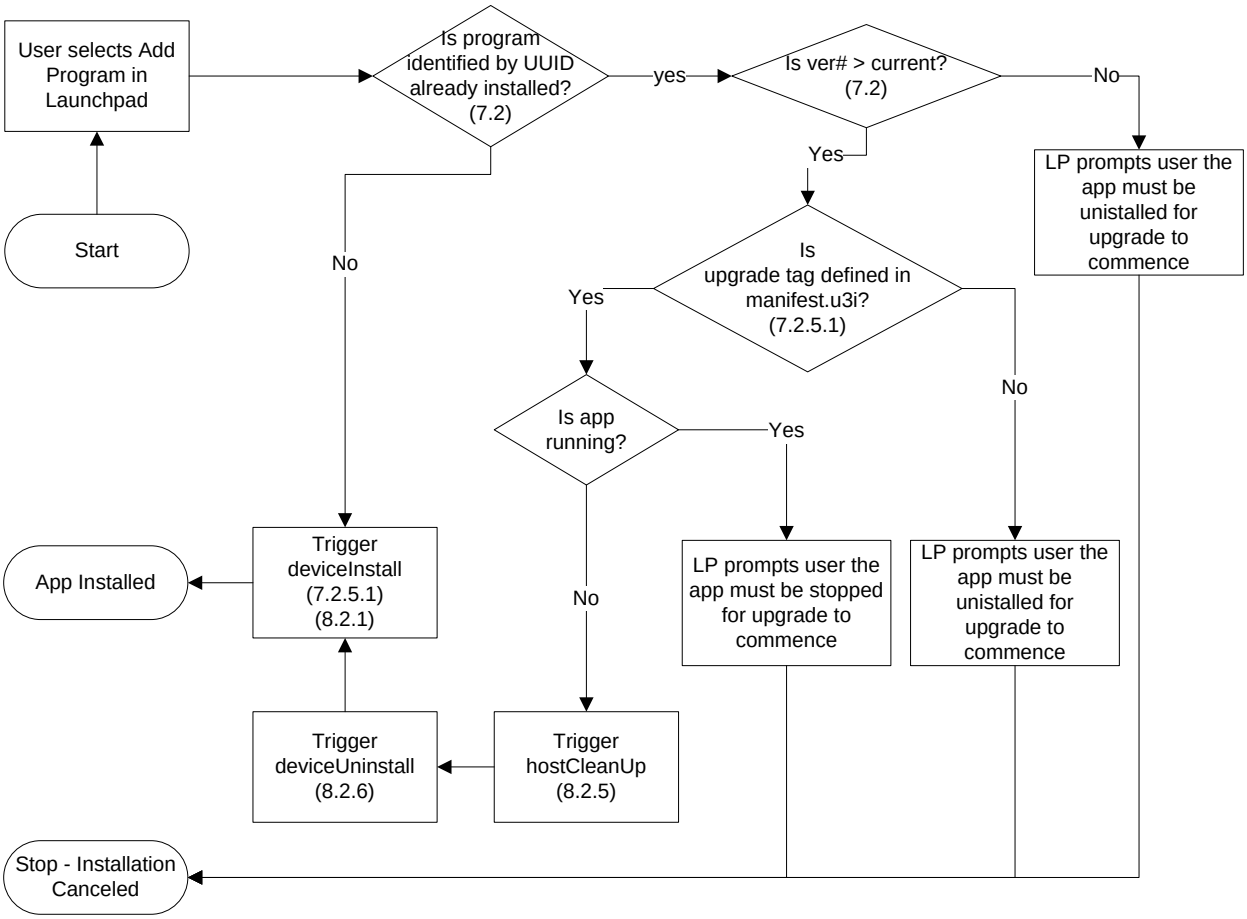


Figure 10 Upgrade Process Flowchart

9. U3 Runtime Variables

The U3 Launchpad creates the runtime environment for a U3 application, namely the temporary directories on the host machine and the U3 device from which the U3 application is executed. The environment variables allow the U3 application to determine the location of the directories from which it runs. The environment variables include the path to the U3 device and the path to the U3 application executable and data files.

The variables described in this section point to the U3 application and U3 device data required for the U3 application to run. Unless otherwise stated, each value is unique for each running U3 application.

A U3 application can only use the variables described below, and cannot set them.

The variables are accessible as follows:

- As runtime environment variables. For example, a C++ application can use the getenv() function to obtain the value of the variable pszDevicePath = getenv("U3_DEVICE_PATH"). In addition, batch files can use environment variables directly. For example, cd /D %U3_DEVICE_PATH%
- As command line parameters for each U3 action. Variable names can be used in the Actions section of the manifest file, in any part of the action definition. Variable names must be surrounded by percentage symbols (%). The U3 Launchpad replaces each variable name with its actual value before the command is executed.

Variables that return a path to a directory do not contain a trailing '\' character, for example, F: and D:\My App\Data.

There are four types of runtime variables:

- Device information
- Path
- Environment information
- General

9.1. Device Information Variables

The following variables provide general information about the U3 device from which the U3 application is running. These values are the same for all U3 applications executing from the same U3 device.

9.1.1. U3_DEVICE_SERIAL

The serial number of the U3 device. This value may be used to uniquely identify or lock the U3 application to the U3 device. For example, 06E08A4153416A58.

9.1.2. U3_DEVICE_PATH

The drive letter or path to the U3 device that is executing the current U3 application (for example, [F:]). The drive letter of the U3 device is assigned automatically by the OS, and this variable reflects this information so that programs can use the U3 device path transparently.

9.1.3. U3_DEVICE_DOCUMENT_PATH

For Version 1.0 of the U3 environment, this value is: `<U3_DEVICE_PATH>\Documents`, the path to the device Documents directory located on the U3 device. Applications can create sub-directories within this directory to store user data files associated with the U3 application. This variable should be used to ensure that the U3 application will support future locations and names of the Documents directory.

Note: This is the location where U3 applications should create sub-directories to store user files.

9.1.4. U3_DEVICE_VENDOR

The vendor name string. For example, USB Corp.

9.1.5. U3_DEVICE_PRODUCT

The product name string. For example, MicroU3 High-Speed.

9.1.6. U3_DEVICE_VENDOR_ID

The string representation of the integer vendor ID value of the device (VID) example 2284 which is equivalent to the hexadecimal vendor ID 0x08ec.

9.2. Path Variables

The variables described in the following sections are provided by the U3 Launchpad to allow the U3 applications to determine the paths to the directories that make up the U3 application's runtime environment. To obtain the path to the U3 device, use the U3_DEVICE_PATH variable. To obtain the path to the device Documents directory use U3_DEVICE_DOCUMENT_PATH.

Note: It is important that U3 applications always use the path variables, and do not calculate paths based upon the current known locations. U3 plans to introduce more advanced devices in the future, with features such as multiple partitions. For this reason, directories that today have adjacent locations may sit on different drives in future versions.

9.2.1. U3_APP_DATA_PATH

For the first generation of the U3 environment, this value is: `<U3_DEVICE_PATH>\System\Apps\{app_unique_id}\Data`

This is an application-specific directory on the U3 device where the U3 application may store general configuration files, such a licenses, user settings, and so on. This information moves with the U3 application from host to host. It is not recommended to store host-specific information or user files in this location. The directory and its

contents are deleted when the U3 application is uninstalled from the U3 device. See Section 6.2, Application Data Directory.

9.2.2. U3_HOST_EXEC_PATH

For the first generation of the U3 environment, this value is:
`%APPDATA%\U3\{device_serial_number}\{app_unique_id}\Exec`

This is the temporary directory on the host machine where the U3 application files are extracted prior to execution. Any DLLs or other files that make up the U3 application can be found relative to this path.

9.2.3. U3_DEVICE_EXEC_PATH

For the first generation of the U3 environment, this value is:
`<U3_DEVICE_PATH>\System\Apps\{app_unique_id}\Exec`

This is an application-specific directory on the U3 device where the U3 application may store infrequently used application files that are not required on the host machine. This information moves with the U3 application from host to host. It is not recommended to store machine-specific information or user files in this location. The directory and its contents are deleted when the U3 application is either upgraded or uninstalled from the U3 device.

9.3. Environment Information Variables

9.3.1. U3_ENV_VERSION

For the first generation of the U3 environment, the value is: 1.0

This is the version of the execution environment in which the U3 application is running. This value defines the list of variables and dynamic paths that are available. Version 1.0 (first generation) supports all variables defined in this section.

9.3.2. U3_ENV_LANGUAGE

This is the current configured language of the U3 Launchpad GUI. This variable is created and set when the U3 application is executed, and does not change later on.

The value provided is the LCID in decimal notation (rather then Hex). See Appendix B, Supported International Location IDs for a list of supported LCIDs and additional information for working with LCIDs.

Note: A possible scenario may occur in which the U3 Launchpad GUI language is set to English when a U3 application is first launched, and changed to French while the U3 application is still running. Since the

variable does not change to reflect the new language that has been set, the U3 application is not aware that the U3 Launchpad language settings have changed.

9.4. General Variables

9.4.1. U3_IS_UPGRADE

Value: true/false

If the current action is part of an application upgrade, the value is set to the string value “true”. This maybe set during an appStop, hostCleanUp, or deviceInstall action. Otherwise, the value is always “false”.

9.4.2. U3_IS_DEVICE_AVAILABLE

Value: true/false

If the action is being performed with the U3 device already removed from the host machine, the value is the string value “false”. This may be set during a hostCleanUp or appStop action. The default value when the U3 device is inserted is “true”. If the U3 device is removed during the action, the value is not updated.

If the value is "false", the U3 application must not attempt to connect to the U3 device using the Device API (DAPI).

9.4.3. U3_IS_AUTORUN

If the U3 application is being run based on an autorun setting, this value is set to true. This is only set for the hostConfigure and appStart actions.

9.4.4. U3_DAPI_CONNECT_STRING

This string is provided by the U3 Launchpad to allow U3 applications to use DAPI. Applications using DAPI require this string to pass to DAPI in order to connect to the U3 device. See the *DAPI Reference Guide* or the example in in the *SDK Developer Guide* for more information.

9.4.5. U3_ENV_SUB_VERSION

A new version tracking system is being introduced with version 1.0.5.17 of the U3 Launchpad. Because there can many versions of the Launchpad that are released with different features that have no impact on how U3 smart applications interact with the U3 framework, a version tracking system isolated from the U3 Launchpad version number itself is being provided. This new version number will be captured in a U3 runtime environment variable with the name

"U3_ENV_SUB_VERSION" and it will have an integer value. Consult the U3 Knowledge Base for updated information.

10. Action Logging

U3 Launchpad version 1.2 can be set to output debug information during application development. The messages output indicate the completion of specific actions. This document explains how to set up U3 Launchpad to output the information and the message format.

10.1. Action Logging Set UP

To enable logging, add the following registry key.

HKEY_CURRENT_USER\Software\MUSK\SDK\DevLog

DevLog should be a DWORD registry key with a value set to 1.

Insert the U3 smart drive with the U3 Launchpad v1.2. You can output the action messages to a file or a debugging tool

10.1.1. Output to a File

The U3 Launchpad default output is the file lplog.txt. It is written to the following directory:

\Documents and Settings\<username>\Application Data\U3\<device number>

This is a temporary file erased during host cleanup. To preserve the file:

- Do a safe eject
- When the "It is safe to remove..." message appears do **not** remove the device.
- Copy lplog.txt to a different host directory.

Once you have copied the file, you can remove the device.

10.1.2. Output to a Tool

Alternatively, you can use a tool such as DebugView to view the logging output. DebugView is a free utility available from SysInternals and can be downloaded by going to the URL:

`http://www.sysinternals.com/utilities/debugview.html`

Start DebugView before inserting the U3 smart drive. The log output then appears on the screen (see Figure 1). You can then also save the output to a file.

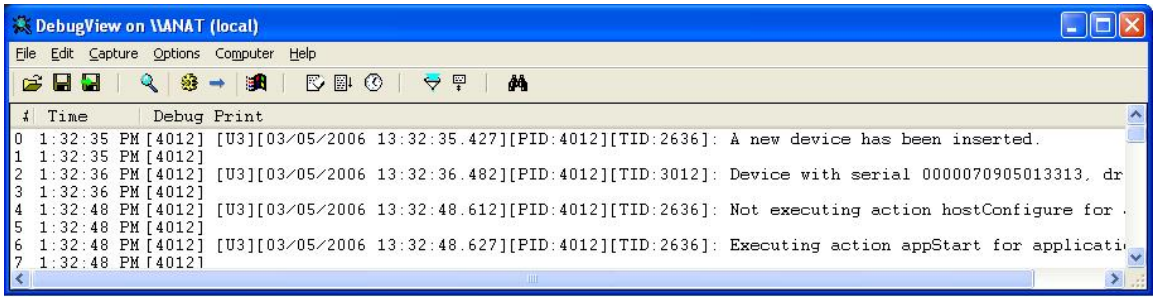


Figure 11 Sample Debug Output

10.2. Message Format

Each message has the following format

[U3][Date][Time][process ID][thread ID]: message

where

- **Date** is in European (dd/mm/yyyy), not American, format
- **process ID** is process identifier used by the OS to identify the process
- **thread ID** is the thread identifier used by the OS to identify the thread
- **message** is a string explaining what event has occurred

Note: Several messages include the device's serial number. The device serial number is also displayed by the U3 Launchpad when you click **About U3 smart drive** in the Help and Support menu.

10.3. Actions Logged

The U3 Launchpad outputs messages indicating device or application status at each phase of device and application installation and removal cycles. The message contents explain what happened.

10.3.1. Device Startup

The following events are reported upon device start up

- Device insertion
- When That Drive Is Connected and Services Become Available
- What Drive Letters Have Been Allocated

Sample Device Insertion Messages

[U3][02/05/2006 12:42:16.177][PID:7800][TID:7848]: A new device has been inserted.

[U3][02/05/2006 12:42:17.128][PID:7800][TID:6756]: Device with serial 0000070905013313, drives F and G, is connected and services are available.

10.3.2. Device Eject

The following events are reported when the device is ejected

- Cleanup Timeout Start/Stop
- Which Applications Have Not Completely Shutdown
- Temporary Host Directories Could Not Be Cleaned Up for Eject
- Improperly Closed Application
- Properly Ejected Device and Cleanup

Sample Device Eject Messages

[U3][02/05/2006 13:52:08.290][PID:7964][TID:6828]: Failed to delete directory "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\67F272E0-0541- 42b5-BDAA-B4D66A3770DE" - The directory is not empty.

[U3][02/05/2006 12:45:08.030][PID:7800][TID:6768]: Eject timeout has been reached - some applications have not been closed or properly cleaned.

[U3][02/05/2006 13:55:28.227][PID:7964][TID:7068]: All applications have been successfully stopped and cleaned before eject timeout was reached.

[U3][02/05/2006 13:55:32.348][PID:7964][TID:6828]: Launchpad is shutting down, all U3 applications are cleaned, so Launchpad is cleaning the root folder on the host.

10.3.3. Application Installation Failure

The following events are output to indicate new application installation failure:

- Bad U3P File Format
- Missing U3P File
- Invalid Manifest Syntax
- Couldn't Clean Out Temp Files

Sample Application Failure Messages

[U3][02/05/2006 14:19:04.925][PID:9236][TID:8804]: Failed to extract archive "G:\System\Apps\67F272E0-0541-42b5-BDAA-B4D66A3770DE\notepad.u3p" - archive does not exist.

[U3][02/05/2006 14:19:04.925][PID:9236][TID:8804]: Failed to extract the manifest folder for application 67F272E0-0541-42b5-BDAA-B4D66A3770DE from "G:\System\Apps\67F272E0-0541-42b5-BDAA-B4D66A3770DE\notepad.u3p" to "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\67F272E0-0541-42b5-BDAA-B4D66A3770DE\Manifest".

[U3][02/05/2006 14:19:04.925][PID:9236][TID:8804]: Failed to load application 67F272E0-0541-42b5-BDAA-B4D66A3770DE - failed to extract application manifest files.

[U3][02/05/2006 14:43:21.113][PID:9964][TID:9152]: Failed to load application manifest file "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\67F272E0-0541-42b5-BDAA-B4D66A3770DE\Manifest\manifest.u3i"

10.3.4. Application Uninstall Failure

The following event is output when there is failure to delete U3 resident directories.

Sample Application Uninstall Failure Messages

[U3][30/08/2006 11:08:12.328][PID:5232][TID:5240]: Failed to delete file "I:\System\Apps\86EB5C19-AB0F-4dd7-BE89-BF96301D35A6\Exec\TestVerb.exe" - Access is denied.

[U3][30/08/2006 11:08:12.359][PID:5232][TID:5240]: Failed to delete directory "I:\System\Apps\86EB5C19-AB0F-4dd7-BE89-BF96301D35A6" - The directory is not empty.

10.3.5. Application Logging

In addition to device and application start and exit failures, U3 Launchpad v1.2 also outputs messages for the following events

- Upgrade state for deviceInstall, hostConfigure and deviceUninstall actions
- Upgrade mode information (add, update, overwrite)
- Autorun state for hostConfigure and appStart actions.
- Path to the action executable, working directory, command line arguments
- PID of action executable and return value of each action executable when it returns
- Warning log entry if appStop returns before appStart.
- Warning log entries if appStart, appStop or hostCleanUp return non-zero values.

Sample Application Startup and Shutdown Messages

[U3][02/05/2006 12:43:06.334][PID:7800][TID:7848]: Not executing action hostConfigure for application 87B9FEC2-C662-4cae-BB1D-BF4355E57346. No command is specified in the application's manifest file

[U3][02/05/2006 12:43:06.334][PID:7800][TID:7848]: Executing action appStart for application 87B9FEC2-C662-4cae-BB1D-BF4355E57346. Command = "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662- 4cae-BB1D-BF4355E57346\Exec\PhotoBackPack.exe", params = "", working dir = "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662-4cae-BB1D-BF4355E57346\Exec".

[U3][02/05/2006 12:43:06.427][PID:7800][TID:7848]: Action command "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662- 4cae-BB1D-BF4355E57346\Exec\PhotoBackPack.exe "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662-4cae-BB1D-BF4355E57346\Exec\PhotoBackPack.exe" created successfully. PID = 6712.

[U3][02/05/2006 12:43:14.543][PID:7800][TID:7848]: Action appStart for application 87B9FEC2-C662-4cae-BB1D-BF4355E57346 has exited with return code (0)

[U3][02/05/2006 12:43:14.574][PID:7800] [TID:7848]: Warning: action appStart for application 87B9FEC2-C662-4cae-BB1D-BF4355E57346 has exited with a non-zero return code (1)

[U3][02/05/2006 13:55:27.977][PID:7964][TID:6828]: Executing action hostCleanUp for application 87B9FEC2-C662-4cae-BB1D-BF4355E57346. Command = "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662-4cae-BB1D-BF4355E57346\Exec\PhotoBackPack.exe", params = "-U3CleanUp", working dir = "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662-4cae-BB1D-BF4355E57346\Exec".

[U3][02/05/2006 13:55:27.977][PID:7964][TID:6828]: Action command "C:\Documents and Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662-4cae-BB1D-BF4355E57346\Exec\PhotoBackPack.exe "C:\Documents and

Settings\Jbruzas\Application Data\U3\0000070905013313\87B9FEC2-C662-4cae-BB1D-BF4355E57346\Exec\PhotoBackPack.exe" - U3CleanUp" created successfully. PID = 9856.

[U3][02/05/2006 13:55:28.133][PID:7964][TID:6828]: Action hostCleanUp for application 87B9FEC2-C662-4cae-BB1D-BF4355E57346 has exited with return code (0)[U3][02/05/2006 13:55:28.227][PID:7964][TID:6828]: Action hostCleanUp for application 67F272E0-0541-42b5 -BDAA-B4D66A3770DE has exited with return code (0)

A. GENERATING A UNIQUE ID FOR A U3 APPLICATION

Microsoft provides GUIDgen.exe, a program that enables developers to generate a GUID (Globally Unique Identifier). The Microsoft Platform SDK and Microsoft Visual Studio include the GUIDgen.exe command line program. The program can also be downloaded from http://www.microsoft.com/downloads/details.aspx?familyid=94551F58-484F-4A8C-BB39-ADB270833AFC&displaylang=en .

If the GUIDgen application was installed as part of a Microsoft development environment, use Start→ Run→GUIDgen to run the application.

Otherwise, download the application package from http://www.microsoft.com/downloads/details.aspx?familyid=94551F58-484F-4A8C-BB39-ADB270833AFC&displaylang=en. The downloaded file is a self-extract exe.

To install GUIDgen:

- Double-click the program icon and select a directory for the extracted files at the prompt that is displayed.

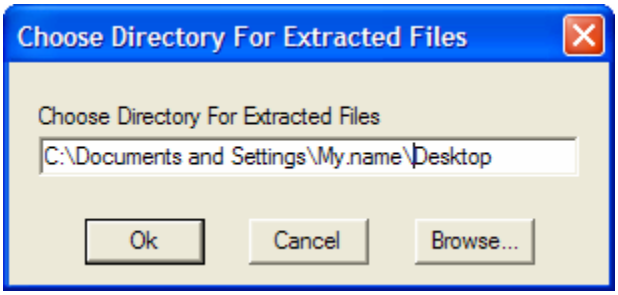


Figure 12: GUID Generation, Choose Directory

- Browse to the required directory and click **OK** to extract the files.
- Navigate to the directory to which the files have been extracted. Locate the file named GUIDgen.EXE.

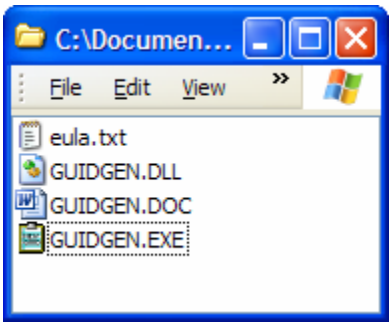


Figure 13: GUID Generation, Extracted Files

To use GUIDgen:

- Run GUIDgen.exe. The Create GUID window is displayed.

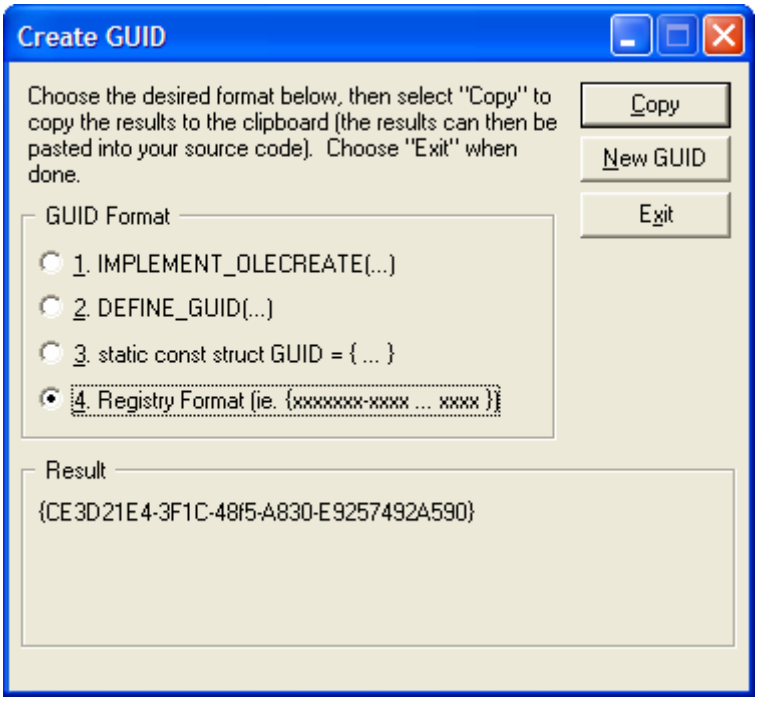


Figure 14: GUID Generation, Create GUID window

- Select option 4. “Registry Format…” To generate a new GUID.
- Click **New GUID** to generate a new GUID. The GUID number is displayed in the Result area.
- Click **Copy** to copy the generated GUID to the clipboard.
- Paste the GUID into the manifest file.
- Delete the parenthesis in the manifest file.

Note: The UUID (GUID) in the manifest file must not be surrounded by parenthesis.

B. SUPPORTED INTERNATIONAL LOCATION IDS (LCIDS)

The following table lists the supported location ID (LCID) values that match each of the languages supported by the Launchpad.

Table 18 List of suported Locations IDs (LCIDs)

Language	Location ID (LCID) decimal value
English	1033
French	1036
Italian	1040
German	1031
Spanish	3082
Japanese	1041

C. Sample Parameter List for a Manifest File

Table 19: Sample Parameter List for a Manifest File

Parameter	Description
Application Name	
Version	
GUID	
Website	
Vendor name	
Vendor URL	
Description	
Additional descriptions	
Default autorun	
Upgrade	
Min free space	
Device Install	
Host configure	
Application start	
Application stop	
Host cleanup	
Device uninstall	
Location: France	1036
Name	
Vendor	
Description	
Location: Italy	1040
Name	
Vendor	
Description	
Location: Germany	1031
Name	
Vendor	
Description	
Location: Spain	3082
Name	

U3 Platform 1.0 SDK
Application Deployment Guide

Parameter	Description
Vendor	
Description	
Location: Japan	1041
Name	
Vendor	
Description	

Return to Overview

U3 Application Deployment Guide
Quick Reference Guide

U3 Package Specification

A U3 Package is an archive file containing all the information and files required to install a U3 application on a U3 device, and run the U3 application from the U3 device.

The U3 Package is a compressed zip file that uses a .u3p extension in place of the .zip extension and is installed on the U3 device using the U3 Launchpad.

The root of a U3 Package may only contain the directories listed below:

Permitted U3 Package Directory Labels

Descriptive Directory Name	Package Directory Label	Description	Extracted To	U3 Reference Variable
Manifest	manifest	Package configuration files	n/a	n/a
Application Data	data	Application persistent configuration files	The device	U3_APP_DATA_PATH
Host Exec	host	Application executable and related files	The host machine	U3_HOST_EXEC_PATH
Device Exec	device	Application auxiliary files	The device	U3_DEVICE_EXEC_PATH

Any files outside of these directories are ignored.

Application Directories Lifecycle

When mobilizing an application, care must be taken when considering the placement of each of the application files in the package directories:

Table 1: Application Directories Lifecycle

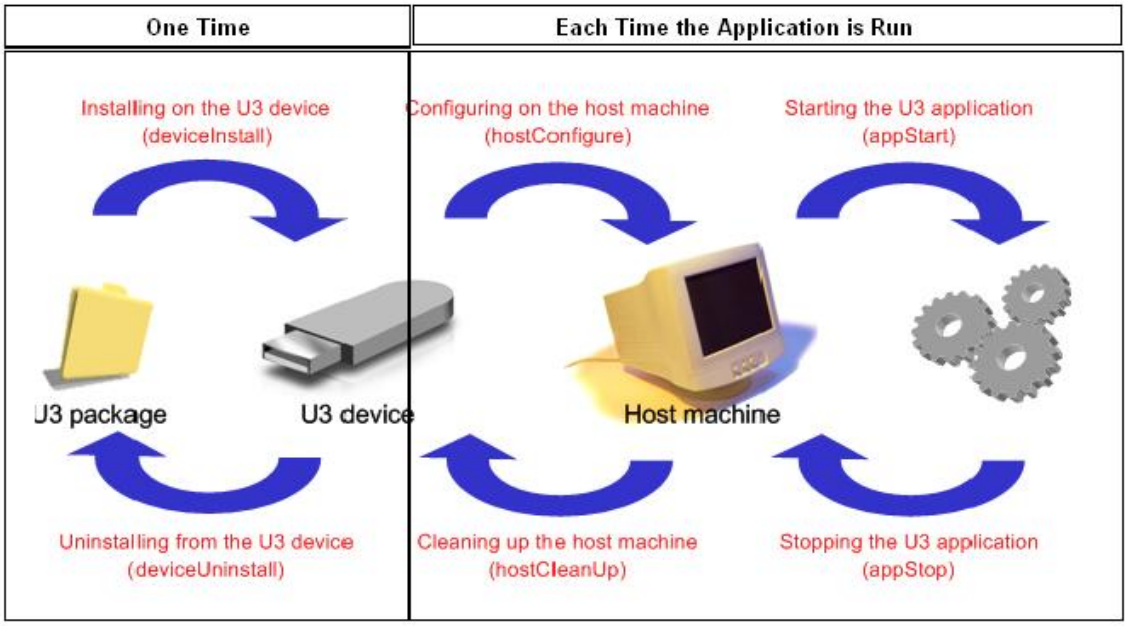
	Package Install	Application Configuration	Cleanup	Application Uninstall	Application Upgrade
Application Data Directory	Extracted			Deleted	Update rules
Device Exec Directory	Extracted			Deleted	Update rules
Host Exec Directory	Extracted	Extracted	Deleted	Deleted	Deleted and re-extracted

Return to Overview

Return to Overview

U3 Application Deployment Guide
Quick Reference Guide

U3 Application Lifecycle



Action Definitions

Action	Required?	LP Action on Return
deviceInstall	No	If non-zero, rollback installation
hostConfigure	No	If non-zero, Application (appStart) is disabled.
appStart	Yes	Must return zero.
appStop	Yes	Must return zero
hostCleanUp	Yes	Must return zero
deviceUninstall	No	If non-zero return, Cancel uninstall

Return to Overview

Return to Overview

U3 Application Deployment Guide
Quick Reference Guide

Runtime variables

- Device:

  U3_DEVICE_SERIAL / U3_DEVICE_PATH / U3_DEVICE_DOCUMENT_PATH

  U3_DEVICE_VENDOR / U3_DEVICE_PRODUCT / U3_DEVICE_VENDOR_ID

- Path:

  U3_APP_DATA_PATH / U3_DEVICE_EXEC_PATH

- Environment:

  U3_ENV_VERSION / U3_ENV_LANGUAGE

- General:

  U3_IS_UPGRADE / U3_IS_DEVICE_AVAILABLE / U3_IS_AUTORUN

  U3_DAPI_CONNECT_STRING

Manifest File Specification

The manifest.u3i file describes the U3 application properties (e.g. name, vendor), installation instructions, and action commands. The manifest file contents are defined using XML tags.

Basic Manifest File Tags

```
<?xml version="1.0" encoding="UTF-8"?>
<u3manifest version="1.0">
<application uuid=" " version=" ">
<icon> </icon>
<name> </name>
<vendor url=" "> </vendor>
<description> </description>
<options>
<autorun/>
<upgrade appData=" " deviceExec=" "/>
<minFreeSpace> </minFreeSpace>
</options>
<i18n>  </i18n>
</application>
<actions>
<deviceInstall
cmd="%U3_HOST_EXEC_PATH%\install.exe" >
        cmdfile=" " </deviceInstall>
<appStop  cmd="%U3_HOST_EXEC_PATH%\main.exe"/>
<appStart  cmd="%U3_HOST_EXEC_PATH%\main.exe">-start
</appStart>
```


Return to Overview

U3 Application Deployment Guide
Quick Reference Guide

```
<hostCleanUp  cmd=" ">-upgrade=%U3_IS_UPGRADE%
</hostCleanUp>
<deviceUninstall workingdir=" " cmd="  ">
</deviceUninstall>
</actions>
</u3manifest>
```

Return to Overview

U3 Manifest Creator User Guide

Version 1.0
June 2006
Revision 1.0

Return to Overview

U3 Manifest Creator User Guide

Copyright and Trademarks

U3 Manifest Creator User Guide

Return to Overview

U3 Manifest Creator User Guide

Table of Contents

Return to Overview

Return to Overview

U3 Manifest Creator User Guide

1. Introduction

The manifest creator is designed to provide a simple GUI for the creation of the manifest file described in the *Application Deployment Guide*. This User Guide focuses on the use the Manifest Creator and not on the contents of the manifest itself.

The manifest creator is a development tool for U3 smart application developers that allows you to:

- Create valid manifest.u3i files
- Edit '.u3i' files
- Check the validity of manifest file inputs against the schema
- Generate a UUID for your application
- Specify U3Action options

The manifest creator has limited support for editing u3i files that do not contain valid XML and manifest files that are not valid when compared to the manifest schema.

For further information on manifest files see the Section 7 of the *Application Deployment Guide*. For additional information on the U3Action options see the *U3Action User Guide*.

2. Menus and Tab Functions

The interface has a standard Windows® application style File Menu for opening, saving files and exiting the application.

You view and edit the manifest file contents using the tabs shown on the main window. The tabs for editing inputs are:

- Manifest Entries
- Required Lifecycle Actions
- Optional Lifecycle Actions
- Internationalization

The tab for viewing current entries is Current Data

You check the validity of your current entries using the File Menu's "Validate Current Entries" option. The shortcut for this is F1. This first checks for the presence and format of certain text entries in the manifest file, and if these are in order, it attempts to validate your current entries against the manifest file schema.

Return to Overview

2.1. Manifest Entries Tab

Use this tab to enter many of the inputs that form the <application> section of the manifest file.

The format for each of these inputs is specified in the *Application Deployment Guide*. You should pay particular attention to the <upgrade> options specified in that document before you select these for your manifest file.

2.2. Lifecycle Actions Tabs

The required lifecycle actions for creating a manifest file using this tool are:

- appStart
- appStop
- hostCleanUp

Important

The manifest creator now includes hostCleanUp as an essential action. This is because of a Launchpad bug in the earliest devices. Consequently, you must define a hostCleanUp action to ensure that clean up takes place correctly when your application runs on these devices.

On the “Required Lifecycle Actions” Tab, select the action you want to edit by clicking on the relevant action name in the “Essential Application Lifecycle Actions” list box.

The optional lifecycle actions are:

- hostConfigure
- deviceInstall
- deviceUninstall

To specify any of these lifecycle actions, check that action’s check box on the checkable list box in the “Optional Lifecycle Actions” Tab.

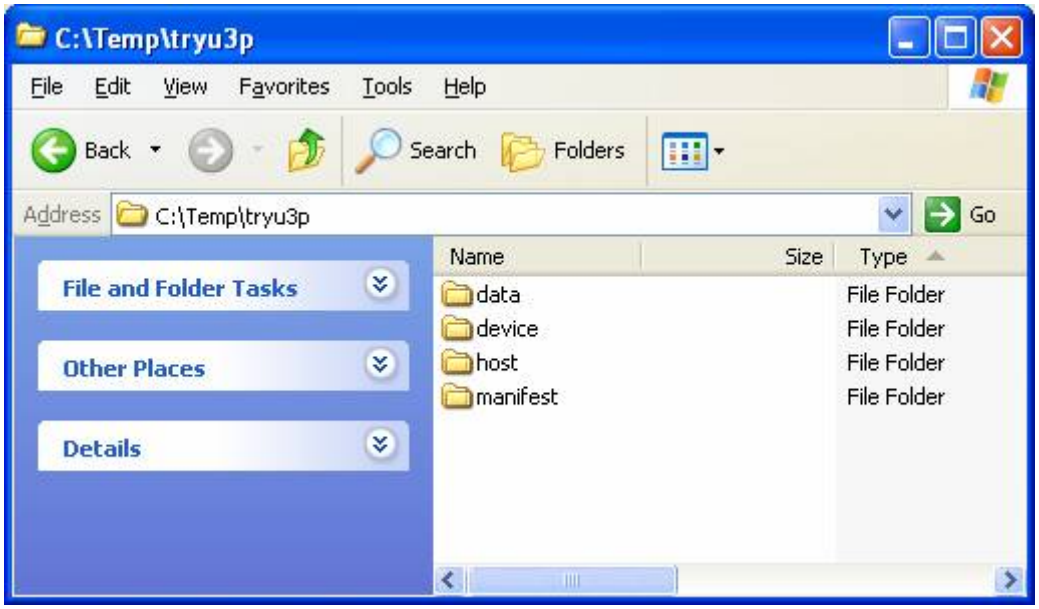
You can edit any highlighted optional action for all actions you have checked in the list box.

2.2.1. “Use U3P project folder” Option

U3 smart applications are by definition mobile applications. The U3 platform requires you to define the location of your U3 application executables in the manifest file in terms of U3 Launchpad environment variables.

The "Use U3P project folder" option helps you to select the executables that you want to be associated with each of the lifecycle actions quickly, and adds the correct environment variables to your

manifest file for you. This option is set as default when you open or create a new manifest file. It is most helpful when you have already placed your project's files in a U3P directory containing the host, manifest, device and data folders as shown below.



If you have opened an existing manifest file, the Manifest Creator assumes that the project folder is that folder above the one in which the manifest file is located. If you create a new manifest file, or wish to change the project folder, use the Browse button to choose a new U3P directory.

Having selected a project folder, you can then choose the action executables for each of the lifecycle actions. For the 'appStop' and 'hostCleanUp' actions, your action executables should be in the host directory to ensure they are available on physical eject. For these actions, if you try to select an action executable that is not in the 'host' folders or one of its subdirectories, you will get a warning message.

For the other lifecycle actions, if you try to select an action executable that is not in one of the four U3P folders or one of their subdirectories, you will get a warning message.

When you select an action executable in one of the four subdirectories, the Manifest Creator will work out the correct path to insert in the manifest file by using a combination of the relative path name and one of the U3P environment variables from the drop down list.

"Use U3P project folder" option unchecked

If you prefer not to specify the U3P directory for selecting your action executables, the "Use U3P project folder" checkbox should be unchecked. In this case, when you select a file, the Manifest Creator removes the root of the selected filename and the rest of the filename you selected goes into the relative path textbox. For each action, select a U3 path environment variable from the drop down list.

The Manifest Creator combines the contents of the "relative path" textbox with the U3 environment variable to form the full command. You should edit the text in the relative path textbox to choose exactly which relative path name you want to be included as part of the "cmd" in the u3i file.

2.2.2. "Use U3Action" Option

If you select "Use U3Action" for a lifecycle action, that lifecycle action is specified in the manifest file using a combination of:

- cmd (in this case it is "%U3_HOST_EXEC_PATH%\U3Action.exe")
- arguments to U3Action (contained between the XML tags)
- The arguments to U3Action.exe consist of:
- Any flags to the U3Action that have been set through the GUI options.
- If specified, the application executable for this stage of the application lifecycle, and its arguments, if any are required.

Your 'appStart' action must have an application executable specified in "Resulting cmd String". This is, after all, the main executable for your application.

For the acceptable combinations and formats of U3Action arguments and text strings, refer to the *U3 Action User's Guide*.

Note: U3Actions for appStart and appStop

If you modify the "Use U3Action" checkbox for either of the appStart or appStop actions, the value you select will be applied to the other action also. This is because use of U3Action for either appStart or appStop alone does not make sense in the context of the U3Action framework.

For more information on how you should combine calls to U3Action for actions in the U3 application lifecycle, refer to the *U3 Action User's Guide*.

If you open an existing manifest file that has U3Action specified for only one of the appStart and appStop actions, the U3Action status of the last of these actions read in from the manifest file will be used for both.

2.3. Internationalization Tab

For each set of <location> settings that you wish to add to the manifest file, you should check the checkbox beside that language name to enable the entry fields for that language. The interface displays the text entry fields for the currently selected language.

2.4. Current Data Tab

This tab allows you to look what the current selections you have made will look like in manifest file format. If you wish to validate the current entries, press F1.

Return to Overview

U3Action User Guide

Version 1.0
June 2006
Revision 1.0

Return to Overview

Return to Overview

U3Action User Guide

Copyright and Trademarks

U3Action User Guide

Return to Overview

U3Action User Guide

Table of Contents

Return to Overview

Return to Overview

U3Action User Guide

1. Introduction

Every U3 smart application must have certain U3 Actions defined to allow the U3 Launchpad to manage the applications through the various U3 runtime phases.

This User Guide introduces an executable module called **U3Action.exe** that performs common U3 actions under the control of the developer via command line options declared in the U3 manifest file.

Sections 2 – 5 correspond to the U3 runtime phase actions. In each section, information for each command line option is given with examples. A table with basic information is provided at the beginning of each section with details where necessary for each command option and general user notes following.

Sections 6 – 8 describe the U3Action commands you use to update an application executable and/or related data which requires the application to be terminated and reactivated. These commands are not a U3 Launchpad-initiated phase and are not specified in your U3P package manifest file. Rather, you execute them directly by your application via the `CreateProcess` API, by a .bat file, or by some other command line mechanism equivalent to typing the command line into cmd.exe and hitting return.

1.1. Disclaimer

U3Action.exe (a.k.a. U3Action) is designed to address the most common application configuration, startup, shutdown, and cleanup activities required by the majority of applications. U3Action also supports a special form of application upgrade cycle in which the application terminates, a helper application updates one or more components, and then the application is restarted so that the new components can be activated.

U3Action does not attempt to address every possible application control scenario. Depending on the complexity of the application, it may be necessary to forgo use of U3Action entirely and modify the control structure of your application specifically for U3 Launchpad events. Conversely, you may wish to rely on U3Action for all signaling and instance control, implementing only the minimum changes necessary within your application to accommodate U3 platform requirements.

More information on U3Action.exe is available in the U3 Developer's Forum Knowledge Base at: https://u3.custhelp.com

1.2. Syntax Rules

U3Action.exe command line switches are not case sensitive.

U3 environment variables **are** CASE SENSITIVE.

U3 manifest file action tags **are** CASE SENSITIVE.

U3Action User Guide

Multiple command options are separated by white space (tabs or spaces, but **not** carriage returns or new lines).

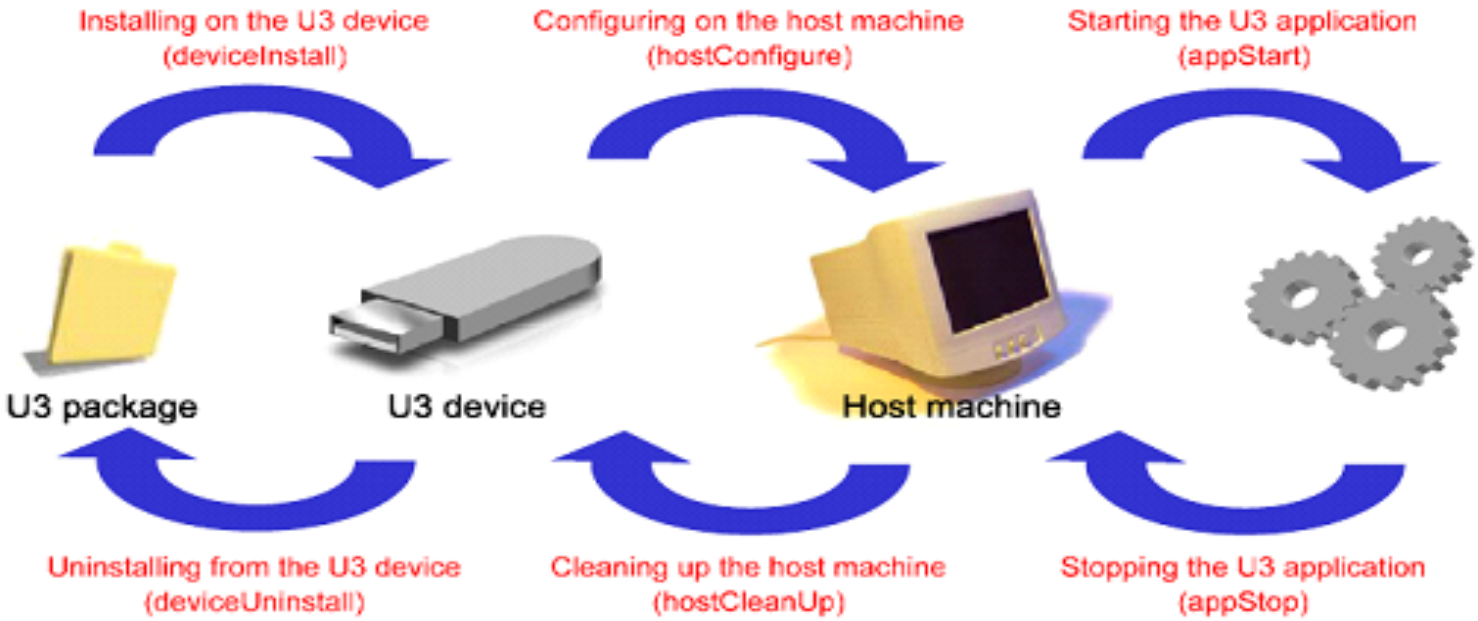
Items are enclosed by square brackets [ ] are optional

Items not enclosed by square brackets are required.

U3Action command line parameters containing spaces must be enclosed by double quotes. In addition, certain command line parameters must also be enclosed in escaped double quotes, for example, the target application executable path.

1.3. U3 Action Symmetry Primer

As a general rule, U3 actions are very symmetrical in nature. Consider the following runtime phase diagram (see section 4.1 of the *U3 Application Deployment Guide* for more details).



Notice the symmetry between the deviceInstall/deviceUninstall, hostConfigure/hostCleanUp, and appStart/appStop phases. Each construction phase (appStart, hostConfigure, deviceInstall) performs a task which is counteracted by its corresponding deconstruction phase (appStop, hostCleanUp, deviceUninstall).

The appStart/appStop phase pair may be the most intuitive example for understanding this relationship: appStart starts the app, appStop stops the app if it's running (e.g. when a device eject event occurs). Similarly, hostConfigure sets up the host machine to support the launch of the

application, while hostCleanUp deletes anything that was installed on the host machine during hostConfigure or appStart.

The fundamental criterion for passing U3 Smart application certification is to leave the host system in the same condition it was in prior to the launch of your application.

For the most part, the construction phases follow developer-specified, application-specific options which control certain behaviors. The deconstruction phases use many of these same options to control how they go about their task. Let's consider appStart and appStop as an example:

- For every executable process instance created by an appStart phase, a process tracking "consort" is created within an internal U3Action data structure. This data structure is shared among all instances of U3Action within the U3 application package, allowing it to track the existence of every process it creates.
- During a Device Eject, the appStop phase of U3Action enumerates through each consort entry, performing whatever process termination operations are specified by the application's manifest's appStop action entry. Using this consort mechanism, U3Action will only terminate processes which it started or adopted during the application's runtime and which were initiated by its parent, the U3 Launchpad process.

In order to monitor and control the various types of spawned processes and action phases, U3Action actually manages several different types of consorts. U3Action's unique consort types share roles and responsibilities with the processes to which each type refers, in much the same way people share different roles and responsibilities within a business organization. There are "manager" consorts, "laborer" consorts, and "restart" consorts to name but a few.

1.3.1. appStop Symmetry Exception

Normally U3 Action phases are serialized; they do not run at the same time. For example, hostCleanUp will not run when hostConfigure is active, deviceUninstall will not run when deviceInstall is active, nor will any combination of unpaired phases.

However, the appStop phase exhibits a slightly different conditional behavior: its runtime execution either overlaps with the appStart execution, or doesn't run at all if no instance of the application is active at the time an Eject event occurs. The following summarizes information about appStop given in section 8.2.4 of the U3 Application Deployment Guide:

- appStop is only called if the process started by the appStart action (as tracked by U3Action and/or the U3 Launchpad) is still running when an eject event occurs. There are several

scenarios where this may not be the case, such as when the user has started an application and subsequently manually stopped it prior to the eject event. Another case is when the application's startup process results in a child process which is completely independent and untraceable by the U3 Launchpad. This situation is highly undesirable, however, as neither U3Action nor the U3 Launchpad can signal the independent process when a device eject event occurs.

- To ensure an appStop action occurs consistently at the proper time to shut down every active instance of an application started by U3Action, the first instance of U3Action for every application will always remain running until all spawned instances of the application terminate. This first instance assumes the primary responsibility for controlling how subsequent U3Action instances behave, and is referred to as the Genesis U3Action.
- It is critical to the proper behavior of U3Action that there be exactly one Genesis instance of U3Action active while the developer's target application (including any of its spawned processes or duplicate instances) is running. Additional instances of U3Action are transient, living only long enough to communicate with the Genesis instance and complete the action phase task for which it was triggered.

1.4. Managing Process Hierarchies

Proper termination of abstract process hierarchies is critical to the proper behavior and successful certification of your application for the U3 platform. U3Action provides two basic mechanisms for managing process hierarchies.

- **-GhostParent**: This U3Action option specified in the appStart and appStop actions controls the way U3Action manages spawned process hierarchies. GhostParent is intended for use with tree-based process control hierarchies, as well as applications which use a "launcher" or "first-run" kicker process to start up the main application instance. The kicker process often terminates shortly after the main process is up and running, leaving the main process with an inactive but valid parent process reference, thus the term "ghost parent." No additional steps are required by the application to inform U3Action of any additional child processes.
- **-Adopt**: An **external** command invoked directly by your application to inform U3Action of any and all additional process instances created by the application. Unlike GhostParent, however, adopted processes do not require a strict process hierarchy of ownership or control. Instead, each adopted process is treated as an equal sibling to other adopted

processes as well as the main appStart application process. The trade-of, however, is that U3Action must be informed of each additional process instance so that it can signal or terminate them during an Eject event, which involves modifying the application source code accordingly.

Normally, U3Action waits for every instance of the target application to exit from each appStart event, exiting itself when the very last instance of the target application has exited. During this time one instance of U3Action is active, managing all process instances. Conversely, GhostParent during appStart causes U3Action to wait on every instance of the target application and their children (if any) before exiting itself..

GhostParent can safely manage one level of descendant processes having an inactive parent process. However, limitations in the host process manager prevent U3Action from tracking all possible spawned process descendant instances (a.k.a. the parent's progeny) automatically. If your application has more complex control needs than provided by the GhostParent feature, consider using the U3Action –adopt option on each application-spawned process, or implementing the process tracking and signaling control directly in your application.

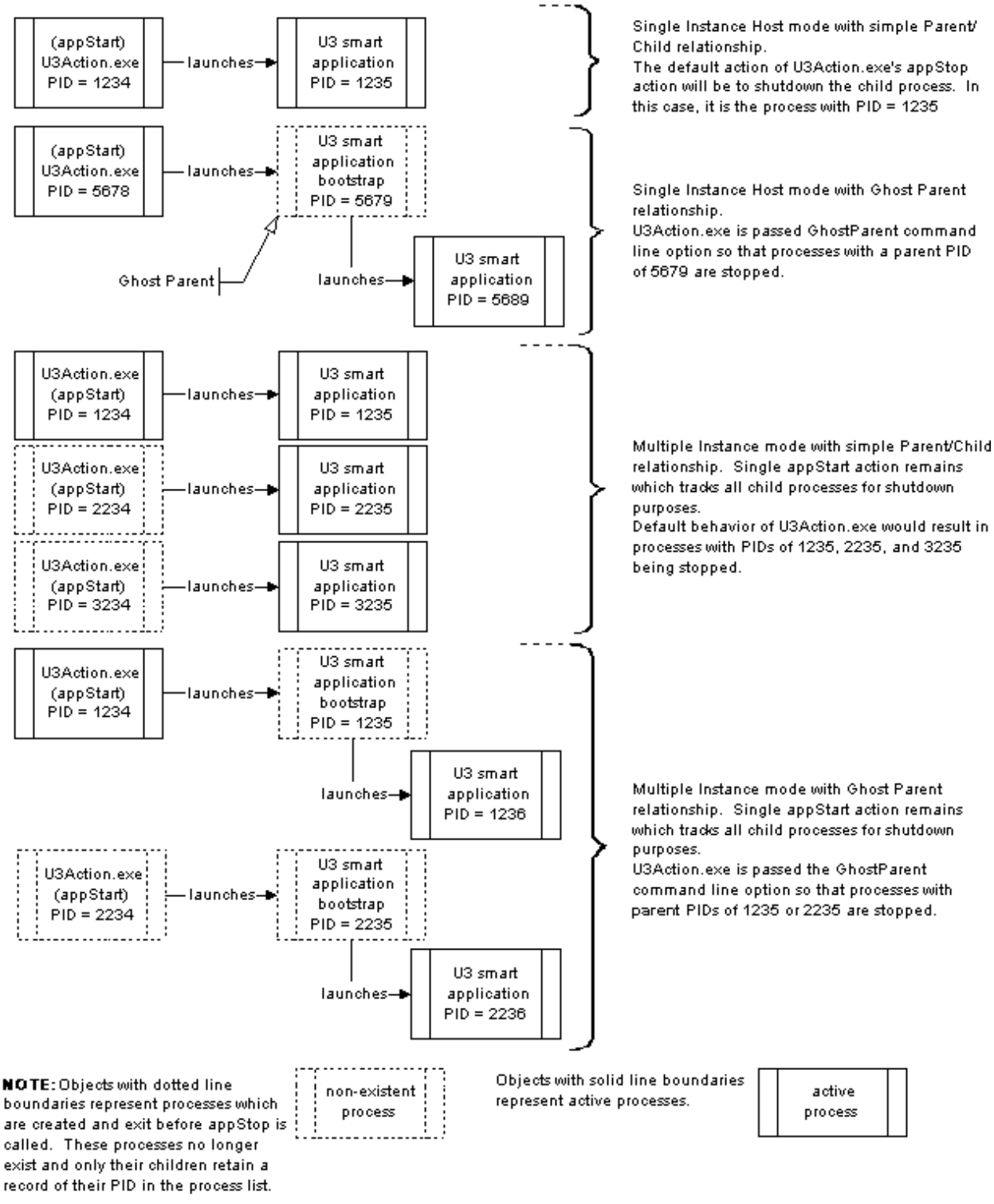
The following diagram shows the relationship of the processes initiated by appStart when using the GhostParent feature. The default behavior, as well as that resulting from use of the GhostParent option, are described in the margin.

Whether or not the GhostParent option is necessary in the appStart and/or appStop command lines depends on several factors:

- The multi instance strategy for the application.
- How you launch the main application executable and its child/parent relationship to the U3Action process which performed the appStart operations.
- How many levels of spawned process descendants can be active and how those descendant processes are signaled for termination during Safe or Physical Ejects.

See the appStart and appStop command descriptions for more information on the use of Ghostparent.

U3Action User Guide



U3Action User Guide

1.5. Registry Portability Considerations

Many applications rely on the host system registry to maintain user preferences or other application-specific data. This creates an application portability problem, as these host registry entries must be installed and removed dynamically each time the U3 smart device resident application is used. In addition, related questions of resolving host registry path conflicts become more difficult to resolve when the same application is installed on or executing from both the host and the U3 smart device. Which version of the registry data should be used? What about conflicting entry formats, foreign Access Control Lists, or other requirements?

To resolve these conflicts, or more accurately to avoid them altogether, U3Action implements a behind-the-scenes automatic registry node installation and archiving mechanism specifically designed for U3 smart devices. The ***only*** change required to the developer's application is to append the U3 smart device vendor number and serial number to the existing registry node path used by the application, and to include the appropriate –SRN and –LR options in your U3 application manifest to activate the registry shuffling feature, as described below.

1.5.1. –SRN *short-reg-path*

This option supports the automatic archival of registry information to standard *.reg text files stored in the U3 smart application's `%U3_APP_DATA_PATH%` directory of the U3 smart device. It may be specified only during the appStart and/or hostCleanUp phases.

The [`-SRN` *short-reg-path*] switch tuple may be specified one or more times consecutively per appStart or hostCleanUp command line. This allows multiple registry nodes or sub-nodes to be saved and/or archived.

short-reg-path refers to a concatenation of the hive name's three letter acronym with the path to your application's node within that hive. The *short-reg-path* for `\Software\MyRegistryNode` in the `HKEY_CURRENT_USER` hive is therefore

"HCU\Software\MyRegistryNode"

The actual name of the registry node to be saved includes the U3 smart device's unique serial number and vendor number. This prevents namespace conflicts with the corresponding registry entries of a host-resident version of your application and with other instances of your application running from other U3 smart devices.

Each copy of the application installed on a U3 smart device uses its own unique copy of the registry node. The only registry-related

change to your application which is necessary, therefore, is to use the U3-formatted version of your registry node's name when running under a U3 smart environment.

The U3-format for automatic registry entries is:

```
your_node_name + dash + vendor_number + dash +
serial_number
```

The registry node may contain sub-node trees, each of which will also be recursively archived.

NOTE: This option is best specified in **both the appStart and hostCleanUp action phases**. When specified in the appStart phase, the specified registry nodes will be written to the U3 smart device **each time** the last instance of the application is terminated, until the U3 smart device is removed and reinserted. When specified in the hostCleanUp action phase, the registry nodes will again be written to the U3 smart device if the device is still available (i.e. a Safe Eject occurred) then they **will always be deleted from the host registry** even if they could not be saved back to the U3 smart device (e.g. a Physical Eject occurred). It is therefore advantageous to specify the –SRN option in appStart, to periodically save the application's registry data back to the U3 smart device.

1.5.2. –LR

This option supports the automatic reload of registry data into the host system registry from standard *.reg text files located in the %U3_APP_DATA_PATH% directory of the U3 smart device. It may be specified only during the hostConfigure or appStart phases.

This switch causes all (if any) previously-archived registry nodes to be loaded into the host system registry.

The –LR option need be specified only once per hostConfigure command line and, unlike –SRN, does not require the short-reg-path of the registry node to be specified.

NOTE: This option is best specified in the hostConfigure action phase. If specified in the appStart phase, all previously-saved registry nodes will be loaded each time the first instance of the application is launched, until the U3 smart device is removed and reinserted. In the hostConfigure usage model, -LR is executed only once per device insertion.

1.5.3. Registry Usage Notes

Currently, only the hives `HKEY_CURRENT_USER` (`HCU`) and `HKEY_LOCAL_MACHINE` (`HLM`) are supported. This is because of

restrictions on hive access under certain host login privileges, for example, Guest Mode logins.

Only the HCU hive can be updated when logged into an account with Guest privileges. See the registry usage notes in the hostConfigure section for more details about this restriction.

Registry node archives self-identify the device on which they reside, but are not truly hardwired to the device. They may be moved from one device to another without harm, but are ignored until they are updated to reflect the vendor number and serial number of the device onto which they have been copied. In other words, they belong to a device and they know which device they belong to, but that information is not recorded within the archive itself.

1.5.4. Examples

```
Environment variable U3_DEVICE_VENDOR_ID = "2284"

Environment variable U3_DEVICE_SERIAL =
"0C016C513380B938"

<hostConfigure cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">

-hostConfigure –LR </hostConfigure>
```

If the U3 smart device was previously safe-ejected, the registry node `\Software\MyRegistryNode-2284-0C016C513380B938` will be automatically reloaded into the `HKEY_CURRENT_USER` registry hive once (and only once) the very first time the application is launched from the U3 smart device.

```
<hostCleanUp cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">

-hostCleanUp -SRN "HCU\Software\MyRegistryNode"

</hostCleanUp>
```

When the last instance of the application terminates, and if the U3 smart device is still physically available at the time hostCleanUp occurs, the registry node `\Software\MyRegistryNode-2284-0C016C513380B938` in the `HKEY_CURRENT_USER` hive will be saved to the `%U3_APP_DATA_PATH%` directory on the U3 smart device. The node will then be deleted from the host system registry.

NOTE: The node will **always** be deleted during hostCleanUp, even if it was not successfully archived to the U3 smart device (for example, a physical eject occurred, thus the storage media is no longer available). It is only when the hostCleanUp action itself does not trigger, either because of an unexpected runtime error or an

incorrectly specified hostCleanUp manifest action, the registry node could ever be left behind.

2. hostConfigure

All uses of U3Action.exe in hostConfigure take the form:

```
<hostConfigure cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">-hostConfigure
[optional hostConfigure group of command line options] [\"Optional
Escaped Application Executable Path\" [optional escaped
application-specific command line parameters]] </hostConfigure>
```

Table 1. hostConfigure Command Line Options

Command	Result / Usage Notes	Example
`-hostConfigure`	Required Sets U3Action in hostConfigure mode	`-hostConfigure`
`-LR`	**Optional** Load Registry loads into the host system registry any **previously-archived** standard *.reg text files found in the %U3_APP_DATA_PATH% directory of the U3 smart device. -LR need only be specified once and unlike -SRN does not require the short-reg-path of the registry node to be specified. It will enumerate through all previously-archived *.reg text files located in the %U3_APP_DATA_PATH% directory automatically. The *.reg file format is the standard text format produced by regedit. Refer to the –LR section above for more information.	`-LR`

[Optional Escaped Application Executable Path [optional escaped application-specific command line parameters]]	**Optional** This is the path to the executable which hostConfigure is to run. U3Action will wait for this application to exit and return its exit code to the U3 Launchpad. If the exit code is zero (meaning no error) the appStart phase will then be initiated, Otherwise, the application is **not** allowed to run. Environment variables available in the runtime environment for the application, including the U3 environment variables, may be specified here as they will be expanded prior to launch of the application. The Escaped Application Executable Path should be enclosed in backslash escaped double quotes (see examples). If there are command line arguments for the application, each command line argument should also be enclosed in escaped double quotes if necessary.	\"%U3_HOST_EXEC_PATH%\My app name.exe\" \"parameter 1\" \"parameter 2\" \"%SystemRoot%\system32\cmd.exe\" \"%SystemRoot%\notepad.exe\" \"%U3_DEVICE_DOCUMENT_PATH%\new file.txt\"

3. appStart

All uses of U3Action in appStart use the following syntax:

<appStart cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">-appStart [optional appStart group of command line options] \"Escaped Application Executable Path\" [optional escaped application-specific command line parameters] </appStart>

Table 2. appStart Command Line Options

Command	Result / Usage Notes	Example
-appStart	Required Sets U3Action in appStart mode	-appStart

U3Action User Guide

Command	Result / Usage Notes	Example

`-SIHost "Executable Image Name"`	Optional
	Will only allow one instance of the application per host machine. If the executable is already running, appStart will attempt to raise its visible windows.
	Otherwise, appStart will start the application as per the remaining command line options.
	The “Executable Image Name” passed in should be exactly as it appears in the system process list.
	The name must be enclosed with double quotes if it contains spaces.
		`-SIHost firefox.exe`
		
		`-SIHost "long executable name with spaces.exe"`

`-SIDevice`	Optional Will only allow one instance of the application per U3 smart device. If the executable is already running from the U3 smart device, appStart will attempt to raise its visible windows. Otherwise, appStart will start the application as per the remaining command line options.	`-SIDevice`
`-GhostParent`	Optional Ordinarily, U3Action waits for every instance of the target application to exit. This option causes U3Action to wait on every instance of the target application ***and*** their children (if any) to exit. This is most often used when the target application is a temporary launcher for the actual application, exiting shortly after the actual application process is up and running. Note: –GhostParent is available both as appStart and appStop options. However, its operation is somewhat different in each. See the description below and in appStop for the details.	`-GhostParent`

U3Action User Guide

Command	Result / Usage Notes	Example
`-SRN "short-reg-path"`	Optional When the last remaining instance of the application terminates, the controlling instance of U3Action activates –SRN processing. Save Registry Node recursively saves the registry node identified by the short-reg-path string **plus an implied U3 smart device specific identifier** to the %U3_APP_DATA_PATH% directory of the U3 smart device. Multiple –SRN options may be specified causing each specified registry node to be saved in an individual binary *.reg file in the %U3_APP_DATA_PATH% directory on the U3 smart device. Registry nodes specified during appStart are **saved but not deleted** from the host system registry. Refer to the –SRN section below for details on how short-reg-paths are mapped to specific U3 smart devices.	`-SRN "HCU\Software\My Registry Node"`

U3Action User Guide

Command	Result / Usage Notes	Example
-LR	Optional When the very first instance of the application launches, the controlling instance of U3Action activates –LR processing. Load Registry loads into the host system registry any previously-archived binary *.reg files found in the %U3_APP_DATA_PATH% directory of the U3 smart device. -LR need only be specified once and unlike -SRN does not require the short-reg-path of the registry node to be specified. It will enumerate through all previously-archived *.reg files located in the %U3_APP_DATA_PATH% directory automatically. The *.reg file format is the binary format produced by system level calls. The *.reg text files produced by exporting from regedit.exe are not supported at this time. Refer to the –LR section below for more information.	-LR
Escaped Application Executable Path [optional escaped application-specific command line parameters]	Required This is the path to the executable which appStart is to run. The very first instance of U3Action appStart becomes a master controller and will wait for this application (and all additional instances of this application and, if the GhostParent option is specified, its children, if any) to exit. Additional appStart actions in a multi-instance launch configuration will spawn new application instances then immediately exit, informing the Genesis instance of U3Action of each new application instance. Environment variables available in the runtime environment for the application, including the U3 environment variables, may be specified here as they will be expanded prior to launch of the application. The Escaped Application Executable Path should	\"%U3_HOST_EXEC_PATH%\My app name.exe\" \"parameter 1\" \"parameter 2\" \"%SystemRoot%\system32\cmd.exe\" \"%SystemRoot%\notepad.exe\" \"%U3_DEVICE_DOCUMENT_PATH%\new file.txt\"

Command	Result / Usage Notes	Example
	be enclosed in backslash escaped double quotes (see examples). If there are command line arguments for the application, each command line argument should also be enclosed in escaped double quotes if necessary.	

3.1. –SIHost

[-SIHost "Application Executable Image Name.exe"]

The -SIHost option enforces a single-instance strategy for an application. When using –SIHost mode, the application executable image will not be launched if an existing version is currently executing when the appStart action occurs. U3Action does not care if the application is installed on and running from the host operating system or from a U3 smart device; only one instance of the application is allowed to execute at a given time.

When the -SIHost option is specified, the next command line parameter must be the executable name including its extension such as “notepad.exe”, “acrord32.exe”, etc. This must be the same name that appears in the Windows Task Manager Processes window (press <CTRL Shift ESC> simultaneously to see this window) or as listed when using the “tasklist” command in a cmd shell or similar utility.

Technically, it is the `PROCESSENTRY32` structure’s `szExeFile` file name field and the `MODULEENTRY32` structure’s `szModule` module name field which are used to determine if the application process is currently executing.

If the application image is already running when the user attempts to start it from any U3 Launchpad, the currently running instance’s main window(s) will be raised to the top of the window order. If no instance is currently executing, one will be started as per the remaining appStart command line arguments.

If neither -SIHost mode nor –SIDevice mode (see below) are specified, then multiple instances of the application are allowed to start. U3Action will track which instances have been started by a specific U3 Launchpad (of a specific U3 Smart device) so that only those processes will be stopped by U3Action during an Eject event.

3.2. –SIDevice

[-SIDevice]

The -SIDevice option provides an alternative single-instance strategy for an application. When using –SIDevice mode, the application executable image will not be launched if an existing version on the **same** U3 smart device is currently executing when the appStart action occurs.

U3Action does not care if the application is installed on or running from the host operating system; it simply ignores the executing host version as if it were running from another U3 smart device.

3.3. –Ghost Parent

When used in appStart, GhostParent tells U3Action to wait on each target process instance AND their children (rather than just the target process instances) to exit. In this case, it attempts to perform a depth-first recursive wait on the "leaf-most" child processes, ending in the target application process, which may or may not still be executing. This guarantees U3Action will wait on all immediate process descendants even if the target application process has terminated.

However, if the descendants had descendants of their own (optionally ad nauseam) any progeny who's ancestor has terminated may cause the tree to appear effectively "pruned" at that point, and none of those orphaned offspring will be waited on.

This is a limitation (or more accurately an optimization) of the process manager. The only way to resolve it is for the target application to control its own process hierarchy more directly. Alternatively, the target application can inform U3Action of new processes using the -adopt command instead of relying on GhostParent behavior.

3.4. Application Executable Path

```
\"Escaped Application Executable Path\" [optional escaped
application-specific command line parameters]
```

The complete pathname to the application must be specified with any necessary command line arguments as per the following rules:

- The complete pathname must be enclosed in backslash-escaped double quotes as follows:
- `\"%U3_HOST_EXEC_PATH%\AppName.exe\"`
- It is **not** necessary to escape backslashes used for directory navigation; only to escape double quotes for parameter itemization.
- Standard command line parameters for the application itself should not require escaping with backslashes, but this is application specific. Do use escaped double quotes to encapsulate any application command line parameters incorporating environment variables whether they are U3 environment variables or not.

3.5. Examples

Case 1: Launch when same application is not running

Assuming that there exists a U3P file with notepad.exe in the host directory, the following will launch a U3 smart instance of notepad if there isn't already a notepad.exe running. It will automatically open foo.txt in the current working directory (%U3_HOST_EXEC_PATH% by default) or prompt the user to create it if it doesn't.

<appStart cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">-appStart -SIHost notepad.exe \"%U3_HOST_EXEC_PATH%\notepad.exe\" foo.txt</appStart>

Case 2: Launch regardless of another running instance

Given the same U3P file from the proceeding example, the following will launch an instance of notepad regardless of any other running instances.

<appStart cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">-appStart \"%U3_HOST_EXEC_PATH%\notepad.exe\"</appStart>

Case 3: Launch only one instance per U3 smart device

Given the same U3P file from the proceeding example, the following will restrict the number of simultaneously active instances to only one per U3 smart device. The host computer itself is treated as if it were another U3 smart device, which effectively means any active instances already executing on the host are ignored.

<appStart cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">-appStart –SIDevice \"%U3_HOST_EXEC_PATH%\notepad.exe\"</appStart>

4. appStop

All uses of U3Action in appStop use the following syntax:

```
<appStop cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">-appStop [optional appStop group of command line options] [\"Optional Escaped Application Executable Path\" [optional escaped application-specific command line parameters]] </appStop>
```

4.1. Eject Modes

There are two types of eject modes: Safe Eject and Physical Eject.

- Safe Eject occurs when the user selects the Eject button on the U3 Launchpad.
- A Physical Eject occurs when the user removes the U3 smart device from the host machine **without** first selecting Safe Eject from the U3 Launchpad.

How U3Action attempts to shutdown applications depends on the eject mode. The following describes U3Action's behavior in these two scenarios:

4.1.1. Safe Eject

Unless Terminate Override mode is specified in the appStop command line, U3Action will send a `WM_CLOSE` window message to either the visible windows of the application as determined by examining process information associated with the process which is being shut down or to a specific window as specified by the manifest's appStop options. U3Action can also send a custom, statically-defined window message to a specific window or to all visible windows of the application. This approach is quite useful in a U3 aware application as a simple method of communicating to the application that it is being terminated by the U3 environment rather than directly by the user or by some other means. This can be especially easy if your window message handler callbacks all invoke a common default handler within your application.

In addition, U3Action appStop can launch a separate executable responsible for messaging whatever components of the application are necessary to shut it down completely. This approach allows applications to interact with the user if necessary to save files or preferences or perform other actions which may require user interaction.

If Terminate Override is specified, U3Action will treat a safe eject event as though it was a physical eject event and terminate the application process immediately “with prejudice.” See the following section for details on the physical eject behavior.

4.1.2. Physical Eject

U3Action will perform a recursive series of `TerminateProcess` calls on each instance of the application and its progeny. This is a very aggressive approach but is the only way to virtually guarantee that all application instances are shut down and cleaned up within the 5 second time period allowed by the U3 Launchpad.

Because it is a physical eject, the host machine must be returned to the state it was in before the U3 smart drive was connected as per the U3 Certification Criteria. In this scenario, the application cannot save information back to the U3 device because the device is physically no longer present.

If the application has defined a hostCleanUp action, and all required executables are located in the host directory of the U3P package, the U3 Launchpad will have copied them to a temporary host execution directory prior to appStart of the application. In this situation, U3Action and these executables are allowed to perform their cleanup activities

even though the U3 smart device is physically no longer available, subject to the 5 second time limit imposed by the U3 Launchpad.

It is strongly suggested that the developer **always** place U3Action.exe in the host directory of the U3P package, even if the developer's application is executed directly from the U3 smart device. This helps to ensure a safe, controlled shutdown of all active applications launched from the U3 smart device by isolating the U3 Launchpad from any indirect effects caused by the developer application's media literally being pulled out from under it as it is executing.

Table 3. appStop Command Line Options

Command	Result / Usage Notes	Example
`-appStop`	Required Sets U3Action in appStop mode	`-appStop`
`-TO`	Optional Activates Terminate Override mode Terminate Override mode will process safe eject events as though they are physical eject events.	`-TO`
`-GhostParent`	Optional Indicates that the process started by U3Action may not be active when an appStop action occurs, and that the process or processes to be stopped may be orphaned descendants of the original application process. Depth-first recursive genocide is performed on the entire process tree, using the appStop termination method specified by the remaining appStop options. Note: –GhostParent is available both as appStart and appStop options. However, its operation is somewhat different in each. See the description below and in appStop for the details.	`-GhostParent`

U3Action User Guide

Command	Result / Usage Notes	Example
`-WinTitle "Window Title"`	Optional This option specifies the title of the specific window (associated with the process being shut down) which is to receive the shutdown message during a safe eject.	`-WinTitle MyTitle` `-WinTitle "My Window Title With Spaces"`
`-WinClassName "Window Class Name"`	Optional This option specifies the class name of the window associated with the process being shut down which is to receive the shutdown message during a safe eject.	`-WinClassName MyClass` `-WinClassName "My Class With Spaces"`

`-WinMsg msg wParam lParam`	Optional
	This option specifies the message to send to the target window specified by -WinTitle and -WinClassName during a safe eject.
	All three sub-parameters must be specified and are of type `DWORD`.
	Any unused parameters must be set to 0.
	Numeric double word values can be specified in either hexadecimal or decimal notation based on the parsing rules defined in the `–WinMsg` section below.
		`-WinMsg 0x10 0 0`
		where
		msg = decimal value `16`
		wParam = `NULL`
		lParam = `NULL`
		`-WinMsg 012 345 0x2a`
		where
		msg = decimal value `12`
		wParam = decimal value `345`
		lParam = decimal value `42`

Command	Result / Usage Notes	Example
[Optional Escaped Application Executable Path [optional escaped application-specific command line parameters]]	Optional This is the path to the executable which appStop is to run just prior to terminating the application. U3Action will wait for this application to exit before initiating its termination sequence. Environment variables available in the runtime environment for the app, including the U3 environment variables, may be specified here as they will be expanded prior to launch of the application. The Escaped Application Executable Path should be enclosed in backslash escaped double quotes (see examples). If there are command line arguments for the application, each command line argument should also be enclosed in escaped double quotes if necessary.	\"%U3_HOST_EXEC_PATH%\My app name.exe\" \"parameter 1\" \"parameter 2\" \"%SystemRoot%\system32\cmd.exe\" \"%SystemRoot%\notepad.exe\" \"%U3_DEVICE_DOCUMENT_PATH%\new file.txt\"

4.2. –GhostParent

When used in appStop, GhostParent tells U3Action to perform depth-first recursive genocide on the target process instances ***and*** their progeny (rather than just the target process instances). For appStop, U3Action finds the deepest "leaf-most" processes and signals or terminates them first before traveling back up the process tree to the "root-most" target process originally spawned by appStart (which may or may not still be running). The algorithm is intended to handle any configuration of child processes, but the order in which sibling processes are signaled is completely random. Also, termination signaling is limited by the same inactive intermediate-ancestor host process manager issue as appStart has when waiting.

4.3. –WinMsg

Message parameter parsing rules: If the first character is **0** and the second character is '**x**' or '**X**' the string is interpreted as an **unsigned hexadecimal integer**. Otherwise, it is interpreted as an **unsigned decimal integer**. Other base value notations such as octal and binary are **not** supported.

U3Action User Guide

5. hostCleanUp

All uses of U3Action in hostCleanUp take the form:

<hostCleanUp cmd="%U3_HOST_EXEC_PATH%\U3Action.exe">-hostCleanUp [optional hostCleanUp group of command line options] [\"Optional Escaped Application Executable Path\" [optional escaped application-specific command line parameters]] </hostCleanUp>

Table 4. hostCleanUp Command Line Options

Command	Result / Usage Notes	Example
`-hostCleanUp`	Required Sets U3Action in hostCleanUp mode	`-hostCleanUp`
`-SRN "short-reg-path"`	Optional Save Registry Node recursively archives the registry node identified by the short-reg-path string **plus an implied U3 smart device specific identifier** to the %U3_APP_DATA_PATH% directory of the U3 smart device. Multiple –SRN options may be specified causing each specified registry node to be saved in an individual binary *.reg file in the %U3_APP_DATA_PATH% directory on the U3 smart device. Registry nodes specified during hostCleanUp are saved to the U3 smart device if it is still available and then **always deleted** from the host system registry. Refer to the –SRN section for details on how short-reg-paths are mapped to specific U3 smart devices.	`-SRN "HCU\Software\My Registry Node"`

U3Action User Guide

Command	Result / Usage Notes	Example
`-hostCleanUp`	Required Sets U3Action in hostCleanUp mode	`-hostCleanUp`
`-SRN "short-reg-path"`	Optional Save Registry Node recursively archives the registry node identified by the short-reg-path string **plus an implied U3 smart device specific identifier** to the %U3_APP_DATA_PATH% directory of the U3 smart device. Multiple –SRN options may be specified causing each specified registry node to be saved in an individual binary *.reg file in the %U3_APP_DATA_PATH% directory on the U3 smart device. Registry nodes specified during hostCleanUp are saved to the U3 smart device if it is still available and then **always deleted** from the host system registry. Refer to the –SRN section for details on how short-reg-paths are mapped to specific U3 smart devices.	`-SRN "HCU\Software\My Registry Node"`

`[Optional Escaped Application Executable Path [optional escaped application-specific command line parameters] ]`	Optional
	This is the path to the executable which hostCleanUp is to run. U3Action will wait for this application to exit before exiting itself.
	Environment variables available in the runtime environment for the application, including the U3 environment variables, may be specified here as they will be expanded prior to launch of the application.
	The Application Executable Path should be enclosed in escaped double quotes. If there are command line arguments for the application, the Application Executable Path should be enclosed in escaped double quotes and each command line argument should also be enclosed in escaped double quotes if necessary.
		`\"%U3_HOST_EXEC_PATH%\My app name.exe\" \"parameter 1\" \"parameter 2\"`
		
		`\"%SystemRoot%\system32\cmd.exe\" \"%SystemRoot%\notepad.exe\" \"%U3_DEVICE_DOCUMENT_PATH%\new file.txt\"`

5.1. Considerations

Due to a bug in many of the earlier U3 Launchpad releases, a hostCleanUp action is **absolutely required** in every U3 platform application, even if there is no work to be done. The simplest solution to this problem is to simply invoke U3Action –hostCleanUp without an Application Executable Path. U3Action will run, realize there's no work to do, and quickly terminate while reporting a successful exit code to the U3 Launchpad to keep things moving.

If the U3P package manifest does specify a custom hostCleanUp Application Executable Path, the path must refer to an executable on the host computer, not on the device. Similarly, U3Action must also be run from the host computer, not from the device. This is necessary because the hostCleanUp phase **may** execute after the U3 smart device is physically disconnected from the host computer and therefore no longer available from which to run the application or U3Action.

The Application Executable Path should therefore include the U3 environment variable %U3_HOST_EXEC_PATH% and the application executable should be placed in the host directory of the U3P package, so that the U3 Launchpad will copy it to a temporary host exec directory before the application is started. This ensures its availability in the event of a physical eject. Similarly, your hostCleanUp application should ***always assume*** the U3 device is **not** available and proceed accordingly.

6. appRestart

The appRestart U3Action feature supports the update of an application executable and/or related data which requires the application to be terminated and reactivated.

If your application implements such a feature, you will need to use the appRestart command. This is necessary because once an application exits, the U3 Launchpad no longer considers the application active and does not attempt to shut it down during a safe eject. Applications which download a new version of themselves from the Internet (or other source) then exit in order to re-launch that newly-downloaded version, now appear to be host-resident versions of the application as far as the U3 Launchpad is concerned. As a result, the freshly downloaded and running application cannot be shut down by the U3 Launchpad. A safe eject is no longer possible. This scenario often results in the unfortunate loss of valuable user data.

By using U3Action –appRestart to invoke an updater application, the U3 Launchpad effectively remains aware of the active instance of the application (even when it's not presently in the host's active process list) and can respond to a Safe Eject event, even during the update cycle, under normal circumstances.

NOTE: This U3Action phase is not invoked by the U3 Launchpad. Rather, the developer's application invokes U3Action.exe directly, asynchronously passing it the

appropriate command line options to launch the executable responsible for the actual update.

This application may be the same application started during appStart but with appropriate command line parameters which instruct it to begin the update process (as opposed to running normally) or it can be a completely separate executable whose sole responsibility is to update the application and/or its related data. It's your choice.

Critical

Do not re-launch your application from your updater utility. You **must** allow U3Action to re-launch your application so that the U3 Launchpad will know to shut it down properly!

Alternatively, if your appStart command line is generic or cannot be properly specified via the U3P package manifest after the update, you may use the U3Action **–adopt** command (see below) to inform U3Action of any and all new process instances (re-) started by your updater application.

This is how the restart procedure works:

- U3Action first terminates all existing instances of the application using the method specified by the appRestart command line options.
- U3Action then launches the updater application and waits for it to exit.
- Once the updater application has exited, the application is launched again using the original appStart command line options, unless the user requested a Safe Eject while the updater application was running.
- If a Safe Eject event was requested while the updater application was executing, the application is **not** restarted to allow the U3 Launchpad Safe Eject to proceed if it's waiting, or attempted again if it previously failed because the updater application continued running beyond the 5 second U3 Launchpad imposed time limit.
- If a Safe Eject event is not requested during the update, the application is restarted and the U3 Launchpad remains aware of the newly launched instance of the application, exactly as if it was launched by the user from the U3 Launchpad in the first place.

It is **not** possible to specify new appStart command line parameters in appRestart at this time.

Important

Do not attempt to invoke U3Action.exe directly using any other action phase directive such as -hostCleanUp or –appStart, etc. These are all reserved for use by the U3 Launchpad and will almost certainly result in wild and unpredictable behavior. Only the **–appRestart**, **-adopt**, and **–command** actions to U3Action are viable outside the U3 Launchpad initiator.

U3Action User Guide

6.1. Options

All uses of U3Action in appRestart use the following syntax:

"%U3_HOST_EXEC_PATH%\U3Action.exe" -appRestart [optional appRestart group of command line options] ["Optional Updater-Application Executable Path" [optional escaped updater-application-specific command line parameters]]

Note that **appRestart is not a U3 Launchpad-initiated phase**. These parameters **are not** specified in your U3P package manifest file. Rather, you execute them directly by your application via the `CreateProcess` API, by a .bat file, or by some other command line mechanism which is effectively equivalent to typing the above into cmd.exe and hitting return.

Table 5. appRestart Command Line Options

Command	Result / Usage Notes	Example
`-appRestart`	Required Sets U3Action.exe in appRestart mode	`-appRestart`
`all appStop command line options (except -appStop of course)`	See appStop for details.	

`Optional Updater-Application Executable Path [optional escaped updater-application-specific command line parameters]`	Optional
	
	This is the path to the executable which appRestart is to run to perform any required updates to the application executable, its data, its preferences, or otherwise. U3Action will wait for this application to exit before executing the original appStart command line again, unless an eject event occurs during the update.
	
	Environment variables available in the runtime environment for the application, including the U3 environment variables, may be specified here as they will be expanded prior to initiation of the
		`"%U3_HOST_EXEC_PATH%\My app name.exe" \"parameter 1\" \"parameter 2\"`
		
		`"%SystemRoot%\system32\cmd.exe" \"%SystemRoot%\notepad.exe\" \"%U3_DEVICE_DOCUMENT_PATH%\new file.txt\"`

	application. The Updater-Application Executable Path should be enclosed in escaped double quotes. If there are command line arguments for the application, the Updater-Application Executable Path should be enclosed in escaped double quotes and each command line argument should also be enclosed in escaped double quotes if necessary.	

6.2. Examples

The following U3Action appRestart command will quit all instances of the application by sending each of its visible windows a `WM_CLOSE` message, then launches `MyUpdater.exe` located in the `%U3_HOST_EXEC_PATH%` directory to perform the update:

"%U3_HOST_EXEC_PATH%\U3Action.exe" –appRestart "%U3_HOST_EXEC_PATH%\MyUpdater.exe"

This version quits all instances of the application by killing the processes directly, then launches `MyUpdater.exe` located in the `%U3_HOST_EXEC_PATH%` directory:

```
"%U3_HOST_EXEC_PATH%\U3Action.exe" -appRestart –TO
"%U3_HOST_EXEC_PATH%\MyUpdater.exe"
```

This version sends a `WM_CLOSE` message to a specific window of the application, then launches `MyUpdater.exe` located in the `%U3_HOST_EXEC_PATH%` directory:

"%U3_HOST_EXEC_PATH%\U3Action.exe" -appRestart -WinTitle "My Window Title" -WinClassName "My Window Class Name" "%U3_HOST_EXEC_PATH%\MyUpdater.exe"

This version sends a custom window message to a specific window of the application, then launches `MyUpdater.exe` located in the `%U3_HOST_EXEC_PATH%` directory, passing it the full path to a text file containing instructions on how to perform the update:

"%U3_HOST_EXEC_PATH%\U3Action.exe" -appRestart -WinTitle MyWindowTitle -WinClassName MyWindowClassName -WinMsg 1 2 3 "%U3_HOST_EXEC_PATH%\MyUpdater.exe" "%U3_DEVICE_EXEC_PATH%\MyUpdateInstructions.txt"

U3Action User Guide

Note that a *binary executable* is **not** required to use the appRestart feature (or for that matter, any other action phase). The following example demonstrates how to use a batch script to perform the application update function. Keep in mind that the command shell interpreter may be disabled on some public host computers, for security or other reasons, so dependence on it may also involve testing for its availability and restricting the functionality of the application if it has been disabled.

Assume the following text resides in a file called `MyBatchUpdateScript.bat` located in the `%U3_APP_DATA_PATH%` directory of the U3 smart device. The batch script uses the value of the first passed parameter as an instruction telling it how to perform the update. In this simple example, passing `–App` tells the script to perform an application binary update, while passing `–Data` tells the script to perform an application data update. This approach is easily extended to be as simple or as complex as required.

In this silly example, we'll just call `notepad.exe` to demonstrate the concept:

```
@echo off

if    "%1" EQU ""       goto InvalidParameter
if /I "%1" EQU "-App"   goto PerformAppUpdate
if /I "%1" EQU "-Data"  goto PerformDataUpdate

:InvalidParameter
echo.What do you want me to do?
pause
exit /b

:PerformAppUpdate
"%SystemRoot%\notepad.exe"
exit /b

:PerformDataUpdate
replace "%U3_APP_DATA_PATH%\NewData\MyData.dat"
"%U3_APP_DATA_PATH%\OldData" /U
exit /b
```

To use the script in an appRestart action, use the following U3Action –appRestart command:

```
"%U3_HOST_EXEC_PATH%\U3Action.exe" –appRestart "%ComSpec%" /C
"%U3_APP_DATA_PATH%\MyBatchUpdateScript.bat" -App
```

Specifying the `%ComSpec%` environment variable ensures the correct version of `cmd.exe` will be launched. The `/C` parameter instructs `cmd.exe` to process `MyBatchUpdateScript.bat` and then exit. U3Action will wait on the

U3Action User Guide

cmd.exe process it starts to finish processing the batch file, but it will **not** wait on any independent child processes which the batch script may spawn, for example, using the `start` script command. The `–Data` option to the script performs a selective `replace` command on `MyData.dat` if the version of the file in the `NewData` folder is newer than the one in the `OldData` folder. adopt

All uses of U3Action with the adopt command action use the following syntax:

"%U3_HOST_EXEC_PATH%\U3Action.exe" -adopt processID [optional list of additional processIDs to be adopted]

Note that **adopt is not a U3 Launchpad-initiated phase**. These parameters **are not** specified in your U3P package manifest file. Rather, you execute them directly by your application via the `CreateProcess` API, by a .bat file, or by some other command line mechanism which is effectively equivalent to typing the above into cmd.exe and hitting return.

U3Action User Guide

7. command

All uses of U3Action with the adopt command action use the following syntax:

"%U3_HOST_EXEC_PATH%\U3Action.exe" -adopt processID [optional list of additional processIDs to be adopted]

Note that **adopt is not a U3 Launchpad-initiated phase**. These parameters **are not** specified in your U3P package manifest file. Rather, you execute them directly by your application via the `CreateProcess` API, by a .bat file, or by some other command line mechanism which is effectively equivalent to typing the above into cmd.exe and hitting return.

Table 6. Adopt Command line options

Command	Result / Usage Notes	Example
`-adopt`	Required Sets U3Action in Process Adoption mode	`-adopt 1234`
`Process ID in string form`	Required The DWORD process ID value may be specified using either decimal or hexadecimal notation.	
`Additional process IDs to be adopted, in string form.`	Optional Additional process IDs to be adopted can be specified. This is more efficient than invoking U3Action –adopt once for each process when multiple processes are being adopted. This option cannot be used to create an initial appStart instance. The U3 Launchpad is required to initiate at least one instance of the application before adoptions are allowed. U3Action will treat all adopted processes as if they were created by U3Action appStart. They will therefore receive (and must obey) the same termination method used during appStop. You may use the –WinTitle, -WinClassName, and/or –WinMsg appStop options to limit termination handling to a subset of all processes being managed by U3Action, as long as all processes related to the U3 smart device receive an appropriate termination signal within the 5 second U3 Launchpad Safe Eject time limit.	`-adopt 1234 567 0x1425`

	Refer to the Safe Eject document section for more information. U3Action -adopt uses the Command command action internally (see below) to perform the adoption function.	

8. command

All uses of U3Action with a command action use the following syntax:

"%U3_HOST_EXEC_PATH%\U3Action.exe" –command \"Escaped Application Executable Path\" [optional escaped application-specific command line parameters]

Note that **command is not a U3 Launchpad-initiated phase**. These parameters **are not** specified in your U3P package manifest file. Rather, you execute them directly by your application via the `CreateProcess` API, by a .bat file, or by some other command line mechanism which is effectively equivalent to typing the above into cmd.exe and hitting return.

Table 7. Command Command Line Options

Command	Result / Usage Notes	Example
`-command`	Required Sets U3Action in Command Action mode	`-command "cmd.exe"`
`Application Executable Path [optional escaped application-specific command line parameters]`	Required This is the path to the executable which U3Action -command is to run. U3Action will wait for this application to exit before exiting itself. This option cannot be used to create an initial appStart instance; the U3 Launchpad must be used to launch at least one instance of the application before command actions are allowed. The application executable will not be included in an appStop termination cycle, however, use of this feature will stall the U3 Launchpad Safe Eject	`\"%U3_HOST_EXEC_PATH%\My app name.exe\" \"parameter 1\" \"parameter 2\" \"%SystemRoot%\system32\cmd.exe\" \"%SystemRoot%\notepad.exe\" \"%U3_DEVICE_DOCUMENT_PATH%\new file.txt\"`

	mechanism until the – command application exits. This is the *fundamental desideratum* of this feature. Environment variables available in the runtime environment for the app, including the U3 environment variables, may be specified here as they will be expanded prior to launch of the application. The Application Executable Path should be enclosed in escaped double quotes. If there are command line arguments for the application, the Application Executable Path should be enclosed in escaped double quotes and each command line argument should also be enclosed in escaped double quotes if necessary.	

Return to Overview

U3 Platform 1.0 SDK
Application Deployment Guide

Document Control Information

Document No: MAN-SDK10R510806-022

	Title	Name	Date
Issued by:	SDK Product Manager	John Bruzas	December, 2006
Edited by:			

Changes in this document:

- Section 3.3 - Added the LP 1.0 v 1.2.
- Section 4.4 – Added Vista considerations.
- Section 6.6 - Described the difference between the Documents folders created by LP 1.0 vs 1.2.
- Section 9.4 – Added U3_ENV_SUB_VERSION.
- Section 10 - created the Action Logging .
- App B: Added Chinese and the LCID number

Return to Overview